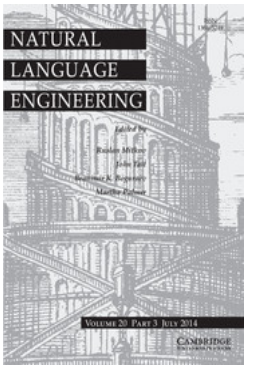

Adaptive graph walk-based similarity measures for parsed text

EINAT MINKOV and WILLIAM W. COHEN

Natural Language Engineering / Volume 20 / Issue 03 / July 2014, pp 361 - 397
DOI: 10.1017/S1351324912000393, Published online: 11 February 2013

Natural Language Engineering **20** (3): 361–397. © Cambridge University Press 2013
doi:10.1017/S1351324912000393

Adaptive graph walk-based similarity measures for parsed text

EINAT MINKOV[1] and WILLIAM W. COHEN[2]
[1]*Department of Information Systems, University of Haifa, Haifa, Israel*
e-mail: einatm@is.haifa.ac.il
[2]*School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA*
e-mail: wcohen@cs.cmu.edu

(*Received 24 May 2012; revised 30 December 2012; accepted 30 December 2012; first published online 11 February 2013*)

Abstract

We consider a dependency-parsed text corpus as an instance of a labeled directed graph, where nodes represent words and weighted directed edges represent the syntactic relations between them. We show that graph walks, combined with existing techniques of supervised learning that model local and global information about the graph walk process, can be used to derive a task-specific word similarity measure in this graph. We also propose and evaluate a new learning method in this framework, a *path-constrained* graph walk variant, in which the walk process is guided by high-level knowledge about meaningful edge sequences (paths) in the graph. Empirical evaluation on the tasks of named entity coordinate term extraction and general word synonym extraction show that this framework is preferable to, or competitive with, vector-based models when learning is applied, and using small to moderate size text corpora.

1 Introduction

Graph-based similarity measures have been successfully applied in recent years to a variety of natural language processing tasks, including word sense disambiguation (Mihalcea 2005; Agirre and Soroa 2009; Navigli and Lapata 2010), sentiment analysis (Kamps *et al.* 2002; Hassan and Radev 2010) and text summarization (Erkan and Radev 2004; Mihalcea and Tarau 2004). Typically in these works, a task-specific similarity measure is derived given a graph in which nodes represent words or word senses, and edges represent a notion of semantic relatedness. The main sources of information used to construct the graph are ontologies, primarily WordNet (Fellbaum 1998). While WordNet, and ontologies in general, are an important resource for lexico-semantic similarity assessments (Toutanova, Manning and Ng 2004; Collins-Thompson and Callan 2005; Hughes and Ramage 2007), they have several inherent shortcomings, mainly ontologies are static and general compared with the dynamic, domain and genre-dependent language usage. For this reason, there is a continuous interest in evaluating lexico-semantic similarity directly from text corpora, which are abundantly available (Lin 1998; Snow, Jurafsky and Ng 2005; Padó and Lapata 2007).

In this paper, we apply graph-based similarity measures directly to text corpora, processing dependency parse trees within a general framework of directed labeled graphs. We propose a graph schema in which word mentions are represented as nodes, linked over labeled dependency edges. The parsed sentence structures are connected in the graph via nodes that denote word types, where a word type is linked to all of its mentions in the corpus. We apply random walks in the graph, following the well-studied Personalized PageRank paradigm (Page *et al.* 1998), to derive a general inter-word similarity measure.[1] In this graph walk paradigm, word types are assumed to be semantically related if they are connected over multiple paths in the graph; in addition, short connecting paths are considered more meaningful.

While graph walks reflect a measure of structural similarity in the graph, learning techniques can be used to further improve the derived corpus-based affinity measure to reflect a particular flavor of word relatedness sought. We consider two learning methods in this work that have been successfully applied in another domain (Minkov and Cohen 2010), namely *edge weight tuning* and *node reranking*. Assuming that the graph edges are associated with a weight according to the relation that they represent, edge weight tuning aims at optimizing the edge weight parameters so that the graph walk process is biased toward relation types that are informative. In the reranking approach, the top nodes ranked according to the graph walk measure are re-ordered using high-level features, including features that describe the *paths* (edge-type sequences) traversed in the walk. As will be demonstrated, path information is highly informative in evaluating word relatedness based on distributional evidence, while the performance of the graph walk and weight tuning techniques is limited due to their local scope.

In this paper, we further describe a non-local *path-constrained graph walk* variant, another approach of learning to rank in this framework. In this method, the random walker is directed to follow paths that lead to relevant nodes with high probability. While reranking is applied to the top nodes ranked by the graph walk, the suggested algorithm incorporates high-level path information already in the graph walk process. We show that this method results in improved performance compared with the unguided walk, as well as with the other learning methods in some cases. In addition, we evaluate the effect of pruning low-probability paths using the path-constrained graph walk variant on performance and computation cost.

The graph representation and the set of learning techniques suggested are empirically evaluated on two different tasks, namely *coordinate term* and *synonym* extraction. In the coordinate term extraction task, we experiment with extraction of named entity classes from text corpora, including *city* and *person* names, given a small set of seed examples. In the second task of synonym extraction, we apply the proposed framework to identify synonymy relations between general English words, where we learn different models for nouns, adjectives and verbs. While the two tasks represent different types of word relatedness, and are typically addressed in the literature separately, we show that the proposed framework can be effectively adapted

[1] Throughout this paper, the term *similarity* will be used to denote general semantic relatedness.

to both types of tasks, demonstrating its generality. Our experiments are conducted using small to moderately sized corpora, where we compare the proposed framework against several vector-based models, and primarily to *dependency vectors* (DV), a state-of-the-art syntactic distributional similarity method (Padó and Lapata 2007). While the dependency vectors assign different, manually tuned and linguistically motivated, weights to the syntactic paths connecting a word to its neighbors, we learn the relative importance of these paths from examples. It is shown that the graph walk-based approach gives preferable or comparable results to the vector-based models in all of our experiments. Unlike vector-based models, which are static, a main advantage of the proposed framework is that it is easily adapted to the specific type of similarity sought.

The primary contributions of this paper are as follows. First, we represent dependency-parsed corpora within a general graph walk framework, and derive inter-word similarity measures using graph walks and learning techniques available in this framework. To our knowledge, the application of graph walks to parsed text in general, and to the evaluated tasks in particular, is novel. Another contribution of this paper is the path-constrained graph walk variant, which is a general technique for learning a global graph walk-based similarity measure in directed and labeled graphs. We discuss the properties and applicability of this method in detail. Finally, we achieve superior or comparable results to a state-of-the-art method on the set of tasks evaluated using small to moderate size text corpora.

The rest of the paper is organized as follows. Section 2 describes prior work related to lexico-semantic similarity assessment and graph-based similarity inference techniques. Section 3 outlines our proposed schema for representing a dependency-parsed text corpus as a graph. The graph walk-based similarity metric is defined in Section 4. In Section 5, we outline the learning methods used, including a formal description of the proposed path-constrained graph walk method; we further comparatively discuss the set of learning methods in terms of expressiveness, impact and applicability. The experimental design and main results on named entity coordinate extraction and synonym extraction tasks are described in Sections 6 and 7 respectively. In Section 8, we discuss and empirically evaluate the effect of design parameters on performance and scalability. The paper concludes with a summary and a discussion of future research directions.

2 Related work

In recent years, graph walks have been widely applied to obtain measures of semantic similarity for Natural Language Processing (NLP) problems. In an early work, Toutanova *et al.* (2004) constructed a directed graph, where nodes represented words and the edges denoted various types of inter-word semantic relations extracted from WordNet. They applied graph walks to infer a measure of word similarity. The semantic similarity scores obtained were used for lexical smoothing for the task of prepositional word attachment. Graph walks over graphs representing word relations extracted from WordNet have since been shown to generate a measure of word similarity preferable to alternative approaches (Hughes and Ramage 2007;

Agirre *et al.* 2009). Graph walk-based similarity measures inferred using graphs that represent WordNet relations have been successfully applied to tasks, including word sense disambiguation (Mihalcea 2005; Navigli and Lapata 2007; Agirre and Soroa 2009), query expansion (Collins-Thompson and Callan 2005), sentiment analysis (Kamps *et al.* 2002; Hassan and Radev 2010), summarization (Erkan and Radev 2004; Mihalcea and Tarau 2004) and more. In this work we apply graph walks to derive *corpus-based* word similarity measures. To our knowledge the approach presented in this paper is novel in representing a corpus as a graph that includes syntactic information (in particular, dependency-parsed text), and is novel in exploring the use of random-walk similarity on such a graph. In addition, while previous works were tailored to extract a particular flavor of word similarity, with the goal of improving the performance of a specific end application, we use learning to tune the generated similarity measure per task.

We note that graphs derived from individual parsed sentences have been widely used. For example, Snow *et al.* (2005) used dependency paths to extract hyponyms from a corpus of parsed text. In particular, they extracted patterns from the parse tree of sentences in which hyponym word pairs co-appeared, and trained a hyponym classifier using these patterns as features. Due to data sparsity, however, the ratio of relevant sentences in the corpus was found to be low. In contrast, we represent text corpora as a connected graph of dependency structures, where the graph walk traverses both within- and cross-sentence paths.

Dependency paths of individual sentences have been used also for general relation extraction. Culotta and Sorenson (2004) explored the detection and classification of instances of relations linking entity pairs, including relations such as 'based-in', 'member' and 'spouse'. In their work, they represent each relation instance as a dependency tree, where for each pair of entities in a sentence, the smallest common subtree in the dependency tree is found. Nodes in the dependency tree are augmented with linguistic and semantic features. Based on the hypothesis that instances containing similar relations share similar substructures in their dependency trees, kernel functions were proposed that estimate the similarity between the subtrees. Empirical evaluation results showed that the tree kernel approach outperformed a bag-of-words kernel, implying that the structural information captured in the tree kernel is useful for the relation extraction problem. Bunescu and Mooney (2005) further observed that the information required to assert a relationship between two named entities in the same sentence is typically captured by the shortest path between the two entities in the undirected version of the dependency graph, and proposed a kernel encoding features over the shortest connecting path. Unlike these works, the approach presented here models many sentences in a connected graph, creating paths between lexical items that do not necessarily co-appear in the same sentence due to shared dependency structures and lexical neighborhoods. Rather than addressing the relation extraction problem as a classification problem, graph walks approach it as a ranking (or, retrieval) problem. While this work focuses on assessing word similarity, the proposed framework may be used for general relation extraction. The rich feature set encoded by the dependency tree kernels can be represented in a graph describing a corpus (for example, using part-of-speech walkable nodes

and edges, WordNet links etc.), or by a node reranking module. General relation extraction, however, is a challenging problem, which we leave for future exploration.

Recently, it has been proposed to extract mentions of terms and the relationships between them from corpora of parsed and semantically processed text, integrating these mentions into a large-scale semantic network. In the generated network, both terms (e.g. 'apple cake', 'apple') and relations (e.g. 'is baked from') are represented as nodes. Once the network is built, spreading activation is used to determine semantic relatedness between terms (Harrington 2010; Wojtinnek, Völker and Pulman 2012). We note that random graph walks and learning techniques have been successfully applied to infer long-range associations between concepts in ontologies extracted from the Web (Lao *et al.* 2012). In this work, we focus on the analysis of lexico-syntactic information, where the original sentence structure is preserved in the graph.

The graph representation described in this paper is perhaps most related to syntax-based vector space models, which derive a notion of semantic similarity from statistics associated with a parsed corpus (Grefenstette 1994; Lin 1998; Padó and Lapata 2007). In most cases, these models construct vectors to represent each word w_i, where every vector element for w_i corresponds to particular 'context' c, representing a count or an indication of whether w_i occurred in context c. A 'context' can refer to simple co-occurrence with another word w_j, to a particular syntactic relation with another word (e.g. a relation of *direct object* with w_j) etc. Given these word vectors, inter-word similarity is evaluated using some appropriate similarity measure for the vector space, such as cosine vector similarity, or *Lin's similarity* (Lin 1998) measure.

In this work, we mainly compare our results against an extended syntactic vector space model called *dependency vectors* (Padó and Lapata 2007), in which word vectors consist of weighted scores that combine co-occurrence frequency and the assumed importance of a context based on properties of the connecting dependency paths. A couple of different weighting schemes are considered in this model: a *length* weighting scheme, assigning lower weight to longer connecting paths; and an *obliqueness* weighting hierarchy (Keenan and Comrie 1977), assigning higher weight to paths that include grammatically salient relations. In an evaluation of word pair similarity based on statistics from a corpus of about 100 million words, this approach was shown to give improvements over several previous vector space models. One important difference between the proposed framework and the DV method is that while the latter is based on manual and fixed choices (regarding the set of paths considered and the weighting scheme), we apply learning to adjust the analogous parameters.

Finally, the framework presented in this paper has been successfully applied to process various tasks in the domain of personal information management (Minkov and Cohen 2010). The graph in that domain described a corpus of semi-structured email messages, where an email header was processed as structured relational data, and unstructured text corresponding to email content was represented in the graph as a bag-of-words. In this work we adapt this framework to parsed text, where word types, word mentions and the syntactic structure binding the word mentions are processed as an entity-relation graph. These word graphs are generally larger, and relevant paths considered are generally longer, compared with the previous work.

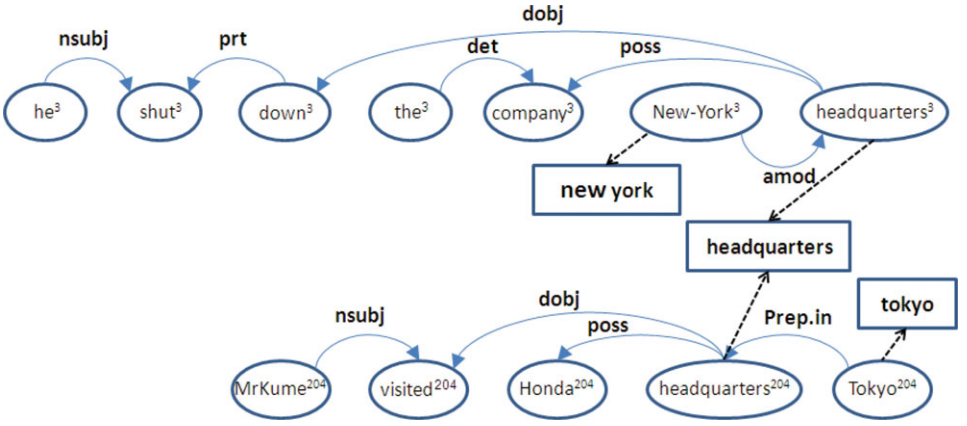

Fig. 1. (Colour online) The suggested graph schema, demonstrated for a two-sentence corpus: 'He shut down the company's New-York headquarters' and 'MrKume visited Honda's headquarters in Tokyo.'

A path-constrained walk (PCW) variant is therefore presented that improves the performance and the scalability in the domain of parsed text. Overall, the framework and methods applied in this paper, including the proposed PCW approach, are applicable to other directed and labeled entity-relation graphs.

3 Graph representation

A typed dependency parse tree consists of directed links between words, where dependencies are labeled with grammatical relations such as *nominal subject*, *indirect object* etc. We suggest representing a text corpus as a connected graph of dependency structures according to the schema shown in Figure 1. The graph shown in the figure corresponds to the dependency analysis of two sentences included in the MUC corpus (see Section 6): 'He shut down the company's New-York headquarters', and 'MrKume himself visited Honda's headquarters in Tokyo's upscale Aoyama district'. The graph displays a simplified version of dependency trees of these sentences, for clarity. As shown, each word mention is represented as a node, associated with the sentence in which it appears.[2] Word mentions are marked as circles in the figure. The 'type' of each word – henceforth a *term* node – is denoted by a square in the figure. (For clarity, only a subset of the graph terms is displayed.) Each word mention is linked to its corresponding term; for example, the nodes 'headquarters^{3}' and 'headquarters^{204}', which represent distinct word mentions, are both linked to the *term* 'headquarters'. For every edge in the graph, we add another edge in the opposite direction (not shown in the figure); for example, an inverse edge exists from 'headquarters^{3}' to 'New-York^{3}' with an edge labeled as *amod-inv*. The resulting graph is highly interconnected and cyclic.

The described schema requires a parsed corpus as its only input. Processing of the corpus into a graph is then straightforward, where term nodes directly correspond

[2] As shown in the figure, sentences are indexed. In practice, a word mention is also marked with its position within the sentence so that each word mention is unique.

to the surface form of the words mentioned. One may refine this representation. For example, term nodes may include part-of-speech tags in addition to lexical values; or, it is possible to replace lexical information with word lemmas. Note, however, that automatically acquired syntactic annotations (including parsing) may be noisy. We follow the schema demonstrated in Figure 1 throughout this paper.

We will apply graph walks to derive an extended measure of similarity, or relatedness, between word *terms*. For example, in Figure 1, starting from the term 'new-york', we will reach the term 'tokyo' via the following path: new-york $\overset{mention}{\longrightarrow}$ new-york^{3} $\overset{amod}{\longrightarrow}$ headquarters^{3} $\overset{as-term}{\longrightarrow}$ headquarters $\overset{mention}{\longrightarrow}$ headquarters^{204} $\overset{prep.in-inv}{\longrightarrow}$ tokyo^{204} $\overset{as-term}{\longrightarrow}$ tokyo. Intuitively, in a graph representing a large corpus, terms that are more semantically related will be linked by a larger number of connecting paths. In addition, short connecting paths may be in general more meaningful (Padó and Lapata 2007). In the next section we show that the graph walk paradigm addresses both of these requirements. Further, different edge types, as well as the *sequence* of edge types traversed, are expected to have varying importance for different types of word similarity; for example, verbs and nouns may be associated with different connectivity patterns (Resnik and Diab 2000). These issues are addressed using learning.

4 Graph walks and similarity queries

This section provides a quick overview of the graph walk-induced similarity measure. The graph walk scheme that we apply is generally known as Personalized PageRank (Page *et al.* 1998; Haveliwala 2002; Agirre and Soroa 2009). In this work, Personalized PageRank scores are approximated through finite graph walks (Toutanova *et al.* 2004; Minkov and Cohen 2010).

Formally, we are given a graph $G =< V, E >$ consisting of a set of nodes V, and a set of labeled directed edges E. Nodes are denoted by letters such as x, y or z, and an edge from x to y with label ℓ is denoted as $x \overset{\ell}{\longrightarrow} y$. Every node x is associated with type $\tau(x)$. The edge labels are limited to a fixed set of possible types.

In our representation of a parsed corpus as a graph, graph node types include *word mentions* and *terms*. The set of edge labels correspond to dependency relations observed, as well as the 'as-term' relation, linking a word mention to the appropriate term. As mentioned above, for every edge in the graph there is an edge going in the other direction, labeled as the inverse relation.

Similarity between two nodes in the graph is defined by a weighted graph walk process, where an edge of type ℓ is assigned an edge weight determined by its type, $\theta_\ell \in \Theta$. (Edge weights can be set uniformly, randomly; or the set of weights Θ can be learned from examples.) The transition probability of reaching node y from node x over a single time step is defined as the weight of the connecting edge, θ_ℓ, normalized by the total outgoing weight over the connecting edges from x to all of its children nodes, $ch(x)$, as follows:

$$Pr(x \longrightarrow y) = \frac{\theta_\ell}{\sum_{y' \in ch(x)} \theta_{\ell'}} \tag{1}$$

Given the underlying transition matrix $\mathbf{M}$, and starting from an initial distribution V_q of interest, Personalized PageRank defines a graph walk scheme, where the distribution of probability scores over the graph nodes in time step d, V_d, is defined as follows:

$$V_{d+1} = \gamma V_q + (1-\gamma)\mathbf{M}V_d \tag{2}$$

The reset factor γ reflects a bias of the random walker interest toward distribution V_q. We will use V_q to represent *a query* of interest; for example, the query $V_q = \{\text{`}new-york\text{'}, \text{`}london\text{'}\}$ designates an interest in items that are similar to the nodes in this distribution, namely city names. Personalized PageRank scores are derived from the corresponding stationary state distribution R. The answer to the query, V_q, is a list of nodes ranked by the scores in distribution R.

The formula of Personalized PageRank graph walk generalizes the more commonly known version of PageRank, as used for webpages (Page *et al.* 1998), in which V_q is uniform over all of the graph nodes, and the stationary distribution reflects query-insensitive centrality scores. It has been shown that the final (stationary) probability distribution of query-sensitive Personalized PageRank walk over the graph nodes, which we denote as R, can be computed as follows (Fogaras *et al.* 2005):

$$R = \sum_{i=1}^{\infty} \gamma^i V_q \mathbf{M}^i \tag{3}$$

According to this Personalized PageRank formula, at each step of the walk i, a proportion γ of the probability mass at every node is emitted. This means that in computing node scores, this probability model applies exponential decay on the distance between the target graph nodes and the query nodes. Focusing on the main paths connecting the graph nodes to a query, we perform a graph walk for a finite number of steps k, approximating R (Toutanova *et al.* 2004). In this multi-step walk, nodes that are reached from the query nodes over shorter paths are assigned a higher score than nodes connected over longer paths. In addition, node scores are monotonic with the number of paths connecting them with the query distribution.

5 Learning

The Personal PageRank metric produces similarity scores that reflect structural information in the graph. It is reasonable to assume, however, that different similarity notions imply varying importance for different link types. In other words, it is unlikely that a single set of parameter values Θ (1) will be best for all queries. Furthermore, the sequences of edge types (*paths*) that are traversed by the graph walk in reaching a target node carry semantic meaning characteristic to the type of relationship between the query and target nodes, where the Markovian graph walk does not model such high-level path information.

We will use learning to adapt the graph walk-based node rankings to reflect a specific flavor of similarity sought. We consider supervised learning settings, where we are given a dataset of example queries and labels indicating (binary) graph node relevancy per these queries. Three different methods of learning to rank graph nodes are applied in this work, including a hill-climbing method that tunes the graph weight

parameters Θ, a node reranking method and a graph walk variant that is constrained to follow meaningful paths. The first two methods have been described in detail elsewhere (Minkov and Cohen 2010). In this section, we provide a brief overview of these methods for completeness. We then motivate and describe the path-constrained graph walk method in detail. This section concludes with a discussion comparing the three learning approaches from the perspectives of utility and applicability.

5.1 Weight tuning

There are several motivations for learning the graph weights Θ in the underlying graph. First, some dependency relations – foremost, *subject* and *object* – are in general more salient than others (Lin 1998; Padó and Lapata 2007). In addition, dependency relations may have varying importance per different notions of word similarity. Weight tuning allows the adaptation of edge weight parameters Θ to the type of similarity sought. Given the adapted edge weights, the graph walk process is biased toward nodes linked over meaningful relations (1).

The weight tuning method implemented in this work is based on an error backpropagation hill-climbing algorithm (Diligenti, Gori and Maggini 2005). The algorithm minimizes the following cost function:

$$E = \frac{1}{N}\sum_{z \in N} e_z = \frac{1}{N}\sum_{z \in N} \frac{1}{2}\left(p_z - p_z^{Opt}\right)^2 \tag{4}$$

where e_z is the error for a target node z, defined as the squared difference between the final score assigned to z by the graph walk p_z, and some ideal score according to the example's labels, p_z^{Opt}.[3] Specifically, p_z^{Opt} is set to 1 in case that the node z is relevant or 0 otherwise. The error is averaged over a set of example instantiations of size N. The cost function is minimized by gradient descent where the derivative of the error with respect to an edge weight θ_ℓ is computed by decomposing the walk into single time steps, and considering the contribution of each node traversed to the final node score. Note that the gradient descent approach is prone to reach local minima, as the target function is not convex (Agarwal, Chakrabarti and Aggarwal 2006). We therefore select the best results out of multiple runs, where edge weights are initialized randomly.

5.2 Node reranking

Reranking of top candidates in a ranked list has been successfully applied to multiple NLP tasks (Collins 2002; Collins and Koo 2005; Shen and Joshi 2005). In essence, discriminative reranking allows the re-ordering of results obtained by methods that perform some form of local search. It is generally assumed that the initial ordering produced by the local search is of high quality so that the top-ranked candidates include relevant answers. The goal in reranking is to improve the initial ordering

[3] For every example query, a handful of the retrieved nodes are considered, including both relevant and irrelevant nodes.

using higher level information. Since features that encode high-level phenomena are typically costly to compute, reranking is applied to a limited number of top-ranked candidates. In our framework, the graph walk serves as a local search procedure that outputs an initially ranked list of nodes, given a large space of candidates corresponding to all of the graph nodes. Only local information is used in the graph walk, as random graph walks are a memory-less Markovian process. We will use reranking to represent nodes in terms of global, high-level connectivity patterns in the graph. The reranking function and features used are described below.

The reranking function: Formally, for every training example i ($1 \leq i \leq N$), the reranking algorithm is provided with an initially ranked list of l_i nodes. Let z_{ij} be the output node ranked at rank j in l_i, and let $p_{z_{ij}}$ be the probability assigned to z_{ij} by the graph walk. Each output node z_{ij} is represented through m features, which are computed by pre-defined feature functions $f_1, \ldots, f_m$. The reranking function for node z_{ij} is defined as

$$F(z_{ij}, \bar{\alpha}) = \alpha_0 \log(p_{z_{ij}}) + \sum_{k=1}^{m} \alpha_k f_k(z_{ij}) \tag{5}$$

where $\bar{\alpha}$ is a vector of real-valued parameters. Given a new test example, the graph walk is first applied to generate a ranked list of nodes, and a small fixed number of the topmost ranked nodes are then reordered by $F(z_{ij}, \bar{\alpha})$, generating a modified ranked list of graph nodes. The parameter weights $\bar{\alpha}$ are learned from the labeled examples provided. We here apply a boosting learning method (Collins and Koo 2005).

Features: We describe a target node z_{ij} in terms of the set of paths traversed from the query distribution V_q in reaching that node by the graph walk process. These path-describing features can be computed in the process of executing the graph walk, or as a post-processing step (Cohen and Minkov 2006). It has been previously shown that reranking graph nodes using path-describing features can improve results significantly (Minkov and Cohen 2010). Specifically, we evaluate the following binary feature templates.

- *Edge label sequences.* Features indicating whether a particular ordered sequence of edge labels ℓ_i (path) occurred within the set of paths leading from the query distribution to the target node z_{ij}.
- *Lexical unigrams.* These features indicate whether a word mention of a particular lexical value t_k was traversed in the graph walk leading to z_{ij}.
- *Source-count*. This feature indicates the number of different source nodes in the query distribution V_q that z_{ij} was reached from. The underlying intuition is that nodes linked to multiple query nodes, where applicable, are more relevant.

Example. Consider the graph depicted in Figure 1, where the query distribution is $V_q = \{$'new york'$\}$. The target node 'tokyo'[4] is described by the features (denoted as *feature-name.feature-value*): *sequence.amod.prep.in-inv* (where the edges *mention* and *isa* are omitted for brevity) and *lexical.headquarters.* (The source-count feature is

[4] This node will be reached if the graph walk is at least six steps long.

degenerate in this case as the query distribution is point-wise.) Another potential target node, 'Honda', is reached from V_q over a different edge sequence, and will be described by the features *sequence.amod.poss* and *lexical.headquarters*. If the reranker is provided with this example query, along with labels indicating that 'tokyo' is a correct response for the query, whereas 'honda' is an incorrect response, it will learn a model that associates the features observed for 'tokyo' and other correct target nodes with high node relevancy and *vice versa*.

5.3 Path-constrained graph walk

While node reranking allows the incorporation of high-level features, it is a costly procedure that is typically applied to a small set of the top candidates retrieved by the local search procedure (aka, the graph walk). Reranking performance is therefore bound by the quality of the initially ranked list of candidates. For this reason, it is desirable to incorporate useful high-level information earlier in the graph walk process. We propose a variant of a graph walk, which is *constrained* by high-level path information. Assume that preliminary knowledge is available that indicates the probability of reaching a relevant node after following a particular edge-type sequence (path) from the query distribution V_q. We are interested in directing a random walker to follow paths for which this probability is high. In the proposed PCW variant, this is achieved by replacing the fixed edge weights Θ with edge weights that are conditioned on the history of the walk. As will be shown, this results in accuracy gains, where paths that lead mostly to irrelevant nodes are degraded in the graph walk process. In addition, it is straightforward to apply a threshold to prune paths with low estimated probability of reaching a relevant node in the walk to improve both qualitative performance and efficiency.

This section describes the path-constrained graph walk variant. The algorithm includes two main components. First, it addresses the evaluation of dynamic edge weights that are conditioned on the history of a walk, based on training examples. The space of path histories is $|\Theta|^K$ for finite graph walks of length K so that a compact representation is required. To this end, a path tree is constructed based on training examples in which each node denotes a unique walk history, and leaf nodes denote full paths observed. Every leaf node in the path tree is associated with the probability of reaching a relevant target node following the underlying edge sequence. These probabilities are propagated in the tree to obtain estimates of edge weights conditional on the history of the walk. The second component of the algorithm adapts the random walk algorithm to consider path history. We represent the nodes traversed as a set of node pairscomprising the graph node and the corresponding vertices in the path tree. The outgoing edge weights from each node pair are estimated according to the respective vertex in the path tree. Next we describe in detail how we construct a path tree from labeled examples, and outline the path-constrained graph walk variant.

5.3.1 The path-tree

We are given a training set of labeled example queries. An initial graph walk of length K (using fixed edge weights Θ) is performed to generate a ranked list of

nodes per query. Let a path p denote a sequence of $j \leq K$ edge types. For each training example i ($1 \leq i \leq N$), we recover all of the full connecting paths leading from each query node to the top M (correct and incorrect) nodes. We consider only acyclic paths.[5] Let C_p^+ be the count of a path p within the paths leading to correct nodes over all example target nodes $N \cdot M$; and similarly, let C_p^- denote its count within the paths leading to the negatively labeled nodes in the example node set. The full set of paths observed can be represented as a tree. (The conversion to a tree is straightforward, where identical path prefixes are merged.) The leaves of the tree correspond to the paths traversed to a target node and are assigned a Laplace-smoothed probability:

$$Pr(p) = \frac{C_p^+ + 1}{C_p^+ + C_p^- + 2} \tag{6}$$

$Pr(p)$ is a (smoothed) maximum likelihood estimate of the probability of reaching a correct node following p, based on observed examples.

While the probability of reaching relevant nodes following various paths over a random walk of length K is estimated directly from data, we are interested in assessing conditional typed edge weights that guide the random walk process in relevant directions, reflecting the probability of reaching a correct target node given partial walk history (corresponding to a path prefix). We approximate the edge weights conditioned on walk history by propagating the path (leaf) probabilities backwards in the path tree to all tree vertices, applying the MAX operator. This approximation provides an upper bound of the downstream probability of reaching a correct response once the walk is fully executed.[6]

Example. Consider the graph shown on the left part of Figure 2. Suppose that the topmost node in the graph (in black) is submitted as a query V_q, where the two nodes at the bottom correspond to relevant (on the left, in blue) and irrelevant (on the right, in red) responses. Assuming a graph walk of three steps or more, there are three acyclic paths overall, including two unique paths that lead to the relevant node:

$$\xrightarrow{\ell}\ \xrightarrow{m}\ \xrightarrow{k}\ \times 2$$
$$\xrightarrow{\ell}\ \xrightarrow{n}\ \xrightarrow{k}\ \times 1$$

The irrelevant node is reached via five paths overall, consisting of three unique paths, as follows.

$$\xrightarrow{\ell}\ \xrightarrow{n}\ \xrightarrow{k}\ \times 2$$
$$\xrightarrow{\ell}\ \xrightarrow{m}\ \xrightarrow{n}\ \times 2$$
$$\xrightarrow{m}\ \xrightarrow{\ell}\ \times 1$$

[5] In our experiments, we found that better performance is obtained if $C(p)$ are evaluated using acyclic paths only, where no node is revisited . Cycle-free network similarity has been discussed in related research (Koren, North and Volinsky 2006).

[6] Interestingly, in reinforcement learning an agent also selects the step that maximizes the future reward in its path to a goal.

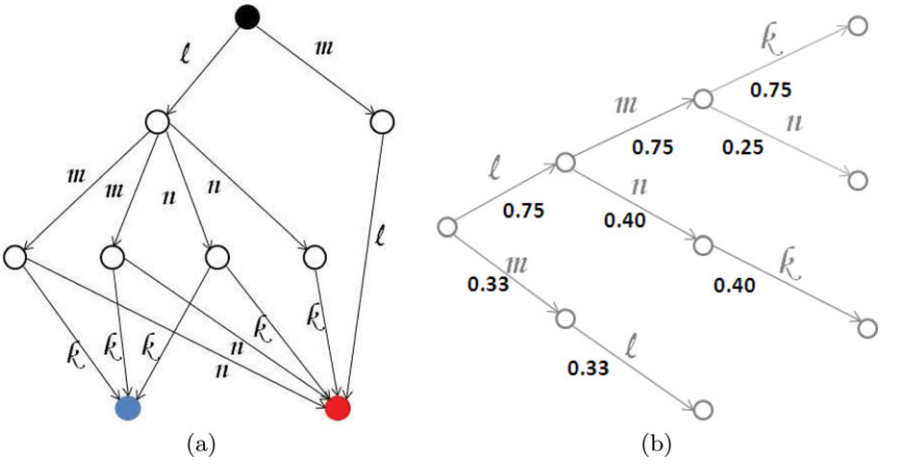

Fig. 2. (Colour online) The toy graph on the left (a) is annotated with a query node (in black), as well as a correct answer node (left bottom node, in blue), and an irrelevant node (right bottom node, in red). The path tree constructed based on this example is shown on the right (b), including the computed conditional typed edge weights. The graph and path tree include the edge labels ℓ, m, n and k.

These path counts are summarized as follows:

$$\begin{aligned} &< C^{+}_{\ell.m.k} = 2, C^{-}_{\ell.m.k} = 0 > \\ &< C^{+}_{\ell.n.k} = 1, C^{-}_{\ell.n.k} = 2 > \\ &< C^{+}_{\ell.m.n} = 0, C^{-}_{\ell.m.n} = 2 > \\ &< C^{+}_{m.\ell} = 0, C^{-}_{m.\ell} = 1 > \end{aligned}$$

In this toy example, the count statistics indicate that the path $\ell.n.k$ is relatively 'noisy', in the sense that this path reaches incorrect responses with high probability (twice out of three times observed). Similarly, the paths $\ell.m.n$ and $m.\ell$ lead to irrelevant nodes only in the observed example. In practice, considering cumulative path counts over a set of example nodes, path count statistics are expected to represent general phenomena in the graph. We wish to assign lower walk probability along noisy paths compared with paths that lead mostly to nodes judged as relevant.

The path tree constructed based on this example is shown on the right part of Figure 2. The conditional typed edge weights are displayed next to each edge in the path tree. According to the computed path counts and (6), the leaf probability of the path $\ell.m.k$ is estimated at 0.75, and at 0.25 for the path $\ell.m.n$. Assuming that the path prefix $\ell.m$ has been traversed in the walk, the outgoing edge weights according to the path tree are $\theta(k|\ell \rightarrow m) = 0.75$ and $\theta(n|\ell \rightarrow m) = 0.25$. Accordingly, given that only an edge of type ℓ was traversed, the weight of an edge of type m is estimated as the maximal probability estimate of the corresponding downstream leaves in the path tree, i.e. $\theta(m|\ell) = \text{Max}(\theta(k|\ell \rightarrow m), \theta(m|\ell \rightarrow m)) = 0.75$. Similarly, at the root of the tree, the computed edge weights are estimated at $\theta(\ell|.) = 0.75$ and $\theta(m|.) = 0.33$. We assume the weights associated with edge types not included in the path tree at a given vertex to be zero.

Algorithm 1 Pseudo-code for path-constrained graph walk

Given: graph G, path-tree T, query distribution V_0, walk length K

Initialize: for each $x_i \in V_0$, assign a pair $< root(T), x_i >$

Repeat for steps $k = 0$ **to** K**:**

For each $< t_i, x_i > \in V_k$:

Let L be the set of outgoing edge labels from x_i, in G.

For each $l_m \in L$:

For each $x_j \in G$ s.t., $x_i \xrightarrow{l_m} x_j$, add $< t_j, x_j >$ to V_{k+1}, where $t_j \in T$, s.t. $t_i \xrightarrow{l_m} t_j$, with probability $Pr(x_i|V_k) \times Pr(l_m|t_i, T)$. (The latter probabilities should be normalized with respect to x_i.)

If t_i is a terminal node in T, emit x_i with probability $Pr(x_i|V_k) \times Pr(t_i|T)$.

5.3.2 A path-constrained graph walk

The *path-constrained* graph walk adheres both to the topology of the graph G and the path tree T. Walk histories of each node x visited in the walk are compactly represented as pairs $\langle t, x \rangle$, where t denotes the relevant vertex in the path tree. This means, however, that if x was reached via n different paths, it would be represented using n node pairs.

Example. Assume that a walk is performed using the path tree shown in Figure 2. If the graph displayed in Figure 2 is used, then after one walk step, the walk distribution would include a couple of node-history pairs: $\langle T(\ell), x_1 \rangle$ and $\langle T(m), x_2 \rangle$. Suppose that another graph node x_3 (in a hypothetical graph) is reached in the next walk step from both x_1 and x_2; in this case, node x_3 would be represented by multiple node pairs, including $\langle T(\ell \rightarrow n), x_3 \rangle$ and $\langle T(m \rightarrow \ell), x_3 \rangle$.

Given the path tree vertex indicated in each node-history pair, the weights of outgoing edges are retrieved from the path tree. These weights are normalized at each node traversed to generate local transition probabilities (following (1)).

Example. Consider the node pair $\langle T(\ell), x_1 \rangle$, where the outgoing edges from node x_1 are of types m, n and k. Assuming that the path tree shown in Figure 2 is provided, the edge weights for this node-history pair are $\theta(m|\ell) = 0.75$ and $\theta(n|\ell) = 0.4$. (We set $\theta(k|\ell) = 0$ because the sequence $\ell \rightarrow k$ is not a path prefix in the path tree.) Given these conditional edge weights, the graph walk proceeds according to its original schema.

The pseudo-code for the path-constrained graph walk is given in Algorithm 1. Note that the number of nodes effectively processed in the path-constrained graph walk is larger relative to an unconstrained walk, as a graph node may be processed multiple times, per walk history. On the other hand, paths in the graph that are not represented in the path tree are pruned. (It is possible, of course, to assign a small probability to previously unseen paths.) In addition, it is straightforward to discard paths in the path tree that are associated with a lower probability than some threshold. A threshold of 0.5, for example, implies that only paths that led to a majority of positively labeled nodes in the training set are followed. Path pruning

has a direct effect on the time complexity of the walk, reducing the number of nodes and edges visited. This is further discussed in Section 8.

5.4 A comparative discussion

We have outlined different approaches for learning to rank graph nodes given labeled examples and initial rankings generated using graph walks. We now discuss the strengths and weaknesses of these methods with respect to several key criteria, including the scope of information modeled by each approach, potential impact of learning on the graph walk ranking and computational aspects.

Scope. A random graph walk process is strictly Markovian, where the random walker does not 'remember' the history of the walk. Similarly, in learning the graph edge weights using methods such as error backpropagation, the graph walk is decomposed into single time steps, and optimization is performed 'locally'. In contrast, the node reranking and constrained graph walk approaches allow one to exploit global properties of the walk; in particular, both methods can model information about *edge sequences* traversed over multiple time steps in the walk. As will be demonstrated, modeling path information is highly important in estimating semantic relations based on distributional evidence in the proposed word graph. Overall, reranking is the most 'global' method out of the approaches considered. In addition to edge sequences, reranking can incorporate features that describe properties of the nodes traversed in a path, for example, their lexical value. Moreover, information about the *collection* of paths leading to a node may be represented as features in reranking. For example, the *source count* feature denotes the number of different query nodes that link to the target node. Similarly, reranking features can explicitly model the number of paths leading to a node, and other global properties pertaining to node connectivity. Finally, reranking can also model arbitrary domain-specific features, incorporating additional relevant information sources that are independent of the graph walk (Minkov and Cohen 2010). In contrast, the proposed formula of the path-constrained walk models information about the edge sequences traversed, but cannot model neither properties of the nodes traversed, or information at path set level.

Impact. Learning may alter the graph walk-based initial rankings to varying extents. The method of tuning the graph edge weight parameters Θ has relatively limited impact on the rankings generated using the Personalized PageRank graph walk scheme. The reason for this is that Personalized PageRank applies an exponential decay over walk length (3); this makes a node's distance from the query nodes the dominant factor in its assigned score. In other words, a strong bias is applied toward nodes that are linked to the query nodes over short connecting paths. Hence, we observe that edge weight tuning mainly affects the relative rankings of 'competing' nodes residing in similar distance from the query. In contrast, the path-constrained graph walk variant can affect the output rankings to a large extent: nodes that are reached from the query nodes over short but unmeaningful paths will be demoted, or excluded from, the output ranking. Unlike reranking, the path-constrained walk applies to all of the graph nodes. Finally, discriminative reranking can significantly

alter the ranked list output by the graph walk. However, for computational reasons, reranking is only applied to the top candidates (nodes) ranked. This means that reranking performance is bound by the quality of the initially ranked lists. It is therefore desirable to apply reranking in combination with a good initial ranking function. In particular, reranking has been successfully applied in combination with graph walks that use a learned set of edge weights Θ^* (Minkov and Cohen 2010). Similarly, we expect that applying reranking to the top results retrieved using the path-constrained walks may be beneficial in some cases; as discussed before, while both methods incorporate high-level information about the edge sequences traversed, additional information can be incorporated using reranking.

Cost. The learning methods differ in terms of training and run-time requirements. In training, the error backpropagation weight-tuning approach requires re-computing the graph walk in each iteration. In addition, in order to avoid local minima, a recommended strategy is to repeat training multiple times using randomized initial edge weight parameters and to select the overall best learned set of edge weights Θ^*. The training procedure of weight tuning is therefore relatively slow. However, having learned Θ^*, a graph walk is readily applied using the modified edge weight parameters with no overhead during runtime. The reranking approach requires a one-time execution of the graph walk to generate the graph walk rankings. Features describing the top graph nodes retrieved can be derived either during the graph walk (Cohen and Minkov 2006), or as a separate step, using a path unfolding procedure (Minkov and Cohen 2010). While the procedure of learning a reranking function and applying the learned model to other feature vectors is efficient, encoding nodes with their feature values adds processing overhead to query execution. Finally, the path-constrained graph walk approach is simple and fast to train. Like reranking, it requires a single execution of the graph walk for the given example queries, as well as path unfolding, during training. In run time, maintaining node pairs that represent the set of walk histories for each graph node traversed imposes relatively large memory requirements, thus affecting the processing time and scalability of the walk. We discuss and evaluate empirically the effect of the PCW algorithm on scalability in Section 8.2.

6 Extraction of named entity coordinate terms

We evaluate the text representation schema and the proposed set of graph-based similarity measures on the task of *coordinate term* extraction. In particular, we evaluate the extraction of named entities, including *city names* and *person names* from newswire data, using word similarity measures. Coordinate terms reflect a particular type of word similarity, and are therefore an appropriate test case for our framework. While named entity coordinate term extraction is often addressed by a rule-based (templates) approach (Hearst 1992), this approach is mainly appropriate for very large corpora such as the Web, where the availability of many redundant documents allows use of high-precision and low-recall rules. In contrast, in this paper we focus on relatively small corpora. Small limited text collections may correspond to documents residing on a personal desktop, email collections, discussion groups

Table 1. *NE coordinate term extraction: corpus statistics*

Corpus	Words	Mention nodes	Term nodes	Edges	Unique NEs
MUC	140 K	65 K	17 K	244 K	3 K
MUC+AP	2440 K	950 K	80 K	3550 K	36 K

and other specialized sets of documents. The task defined in the experiments is to retrieve a ranked list of city or person names given a small set of seeds. This task is implemented in the graph as a query, where we let the query distribution V_q be uniform over the given seeds (and zero elsewhere). Ideally, the ranked list generated in response to the query will be populated with many additional city, or person, names. We compare graph walks with *dependency vectors* (Padó and Lapata 2007), a state-of-the-art syntactic vector-based model, as well as to a vector-based bag-of-words co-occurrence model, where text is processed as a sequence of words rather than as syntactic structures.

6.1 Corpora and datasets

As the experimental corpora, we use the training set portion of the MUC-6 dataset (MUC6 1995), as well as articles from the Associated Press (AP) extracted from the AQUAINT corpus (Bilotti *et al.* 2007), all parsed using the Stanford dependency parser (de Marneffe, MacCartney and Manning 2006). (Due to lower parsing quality, sentences longer than seventy words were omitted.) The MUC corpus provides true named entity tags, while the AQUAINT corpus includes automatically generated noisy, named entity tags. In constructing the graphs, we represent multi-word named entity expressions using a single node; for example, as shown in Figure 1, the named entity mention *New-York*[3] is denoted by a single node, and so is the corresponding (normalized, low-case) term, *new york*. Statistics on the experimental corpora and their corresponding graph representation are detailed in Table 1. As shown, the MUC corpus contains about 140 thousand words, whereas the MUC+AP experimental corpus is substantially larger, containing about 2.5 million words in total. In processing the corpora into word graphs, 'stop words' (uninformative words such as 'the', 'to' etc.) were eliminated. The processed MUC graph contains about 17 thousand *term* nodes, and 65 thousand *word mention* nodes; the MUC+AP graph contains about 80 thousand *term* nodes and about 1 million *word mention* nodes. While the dictionary used grows moderately with the size of the corpus, the number of word mentions is proportional to corpus size. Finally, the graphs are relatively sparse: there are about three connecting edges per node in the graph.

We generated ten queries, each comprising four city names selected randomly according to the distribution of city name mentions in MUC-6. Similarly, we generated a set of ten queries that include four person names selected randomly from the MUC corpus. Using a set of seeds, rather than a single seed, gives a refined definition of the semantic class sought, and is the typical setting in extracting named entity classes (Wang and Cohen 2007). Another reason for using queries

that include multiple entities is that some entity nodes may be poorly connected in the graph and thus provide weak contextual evidence; for example, minor cities are typically mentioned a small number of times in the corpora so that the set of paths originating from the corresponding terms in the graph, and the number of other city names that can be reached over these paths using finite graph walks, are limited. Given multiple seed entities, it is expected that at least a subset of them would provide useful evidence. For each task, we use five queries for training and tuning and the remaining queries for testing. Note that while the number of test queries is relatively small, all of the city, or person, names in the corpus are considered as correct answers. This means that a large number of positive and negative query-target node similarity pairs are evaluated per query. In the experiments, we perform the same set of queries using the MUC corpus, and the larger MUC+AP corpus, so that performance is assessed on the same queries for different corpora sizes. To allow such a comparison, the smaller MUC corpus was appended to AP so that the same query sets are applicable in both cases.

6.2 Experiments

We evaluated cross-validation performance over the training queries in terms of mean average precision (MAP) for varying walk lengths K. We found that beyond $K = 6$ improvements were small (see also Section 8.1). We therefore set $K = 6$. Weight tuning was performed using the training queries and two dozens of target nodes overall. In reranking, we set a feature count cutoff of 3 to avoid over-fitting. Reranking was applied to the top 200 ranked nodes output by the graph walk using the tuned edge weights. Finally, path trees were constructed using the top twenty correct nodes and twenty incorrect nodes retrieved by the uniformly weighted graph walk. In the experiments, we apply a threshold of 0.5 to the path-constrained graph walk method, eliminating paths associated with lower probability of reaching a correct response from the path tree.

We note that for learning, true labels were used for the fully annotated MUC corpus (we hand-labeled all of the named entities of type 'location' in the corpus as to whether they were city names). However, noisy negative examples were considered for the larger automatically annotated AP corpus. (Specifically, for cities, we only considered city names included in the MUC corpus as correct answers.)

A co-occurrence vector-space model was applied using a window of two tokens to the right and to the left of the focus word. Inter-word similarity was evaluated in this model using cosine similarity, where the underlying co-occurrence counts were normalized by log-likelihood ratio (Padó and Lapata 2007). The parameters of the DV method were set based on a cross-validation evaluation (using the MUC+AP corpus). The *medium* set of dependency paths and the *oblique* edge weighting scheme were found to perform best.[7] We experimented with cosine as well as the Lin similarity measure in combination with the DV representation. In applying the

[7] We used the code from underlying syntactic patterns to the Stanford dependency parser conventions.

vector-space-based methods, we compute a similarity score between *every* candidate entity and each of the query terms, and then average these scores (as the query distributions are uniform) to construct a ranked list. Given the large number of candidates in the MUC+AP corpus (see Table 1), we show the results of applying the considered vector-space models to the top, high-quality entities retrieved from this corpus. Specifically, the set of entities considered includes the union of the top 200 results obtained using the tuned graph walks (i.e. the same set of results that was processed using reranking) per query. Finally, for both the graph walk and the vector-based models, we limit the considered set of candidates to named entities.

6.3 Results

We first report the statistics of within-sentence entity co-occurrences in the experimental corpora for the two tasks. In particular, we consider a simple baseline, where one applies a rule extracting all named entities linked over a *conjunction* relation to either one of the query terms (Roark and Charniak 1998). Applying this approach to the city name extraction task yielded overall precision of 80 percent on the MUC corpus and 62 percent on the MUC+AP corpus. However, recall based on intra-sentence term co-occurrences was extremely low: only 7 percent of the city name mentions in the MUC corpus co-appeared with the query terms within individual sentences, and only about 11 percent of those mentions were linked to the query terms over a conjunction relation. Applying this extraction rule to the *person* name extraction task was ineffective, where the rate of person names co-occurring with the query terms within individual sentences was nearly zero. (Alternatively, one can learn contextual rules for person names extraction (Collins and Singer 1999), but such learning is performed in a bootstrapping fashion rather than in a single learning iteration.) Thus, while applying high-precision and low-coverage rules is effective at Web scale, this approach is not appropriate for the processing of the underlying corpora. In contrast, extracting coordinate terms based on common lexico-syntactic neighborhoods does not require term co-occurrence within individual sentences. We next report our results using the graph-based approaches, comparing them against distributional vector-based models.

Figure 3 gives results for the city name (top) and the person name (bottom) extraction tasks. The left part of the figure shows results using the MUC corpus, and its right part shows results using the MUC+AP corpus. The curves show precision as a function of rank in the ranked list, up to rank 100. For this evaluation, we hand-labeled all the top-ranked results as to whether they were city names or person names. Included in the figure are the curves of the graph-walk method with uniform weights (G:Uw) and with learned weights (G:Lw), graph walk with reranking (Rerank) and a path-constrained graph walk. Also given are the results of the co-occurrence model (CO), and the syntactic vector-space DV model, using the Lin similarity measure (DV:Lin). Performance of the DV model using cosine similarity was found comparable or inferior to using the Lin measure, and is omitted from the figure for clarity.

Several trends are observed in the figure. With respect to the graph walk methods, the graph walk using the learned edge weights consistently outperforms

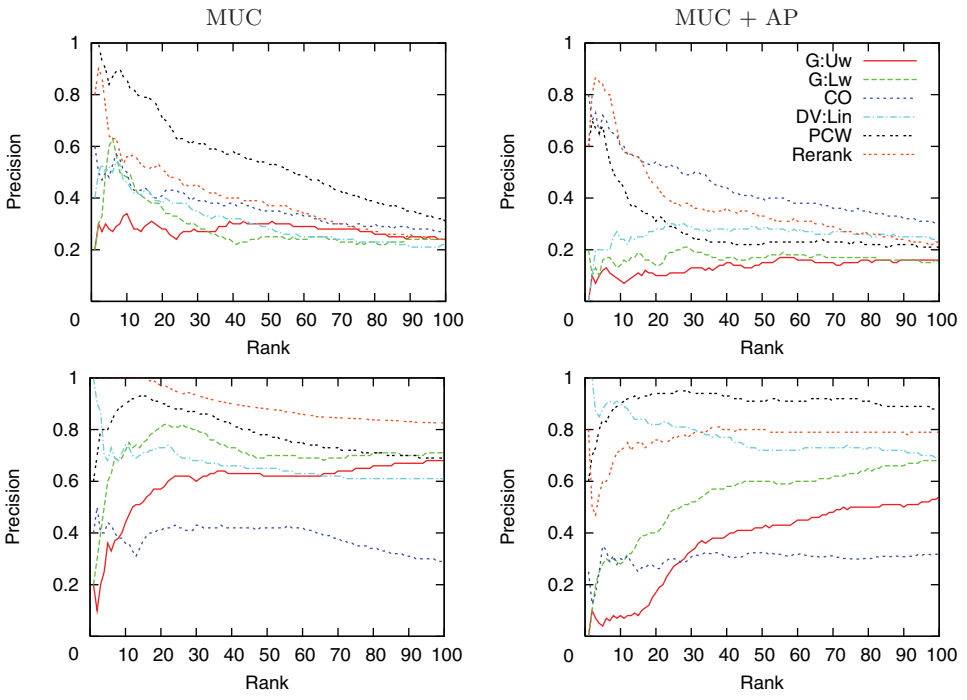

Fig. 3. (Colour online) Test results: precision at the top 100 ranks, for the city name extraction task (top), and person name extraction task (bottom).

the graph walk with uniform weights. Reranking and the path-constrained graph walk, however, yield superior results. Both of these learning methods utilize a richer set of high-level features compared with the graph walk and weight tuning, which can consider only local information. In particular, while the graph walk paradigm assigns lower importance to longer connecting paths, reranking and the path-constrained walker allow the system to discard short yet irrelevant paths and thereby eliminate noise at the top ranks of the retrieved list. In general, the results show that edge sequences carry additional meaning compared with the individual edge label segments traversed.

Out of the vector-based models, the co-occurrence model is preferable for the city name extraction task, and the syntactic DV model gives substantially better performance for person name extraction. In general, we find that city name mentions appear in less structured contexts in the underlying text. For example, the name of the reporting agency and the city in which it resides are usually given in an article's header line, with no meaningful syntactic context. In addition, the syntactic weighting scheme of the DV model is probably not optimal for the case of city names. For example, a *conjunction* relation was found highly indicative for city names (see below). However, this relation is not emphasized by the DV weighting schema. As expected, the performance of the vector-based models improves for larger corpora (Terra and Clarke 2003). These models demonstrate good performance for the larger MUC+AP corpus, but only mediocre performance for the smaller MUC corpus.

Contrasting the graph-based methods with the distributional vector-based models, the difference in performance in favor of reranking and path-constrained walk, especially for the smaller corpus, can be attributed to two factors. The first factor is learning, which optimizes performance for the underlying data. The second factor is the incorporation of non-local information, encoding properties of the traversed paths.

Following is a short description of the models learned by different methods and tasks. Weight tuning assigned high weights to edge types such as *conj-and*, *prep-in* and *prep-from*, *nn*, *appos* and *amod* for the city extraction task. For person extraction, prominent edge types included *subj*, *obj*, *poss* and *nn*. (The latter preferences are similar to the linguistically motivated weights of dependency vectors.) High weight features assigned by reranking for city name extraction included, for example, lexical features such as 'based' and 'downtown', and edge bigrams such as 'prep-in-inv→conj-and' or 'nn-inv→nn'. Positive highly predictive paths in the constructed path tree included many symmetric paths, such as . . . →conj_and-inv. . . →.conj_and. . . , . . . →prep_in-inv. . . →.prep_in. . . , for the city name extraction task.

7 General word synonym extraction

We next report and discuss the results of a set of experiments on the task of synonym extraction using our framework. In this task, given a general word such as 'movie', we are interested in ranking the graph nodes such that its synonyms (e.g. 'film') appear at the top of the ranked list. Downstream applications of synonym extraction include automatic construction of ontologies (e.g. (Lin 1998)), as well as NLP applications, such as summarization (Barzilay and Elhadad 1999), question answering (Lin and Pantel 2001) and textual entailment (Mirkin, Dagan and Geffet 2006). Interestingly, while extensive research has been conducted on extracting measures of general word semantic relatedness from corpora, relatively few works have focused on identifying distinct types of word relatedness, such as hyponymy (Snow *et al.* 2005) or synonymy (van der Plas and Tiedemann 2006). A major strength of the proposed framework is that it can be tuned to reflect special flavors of semantic similarity. Given positive and negative examples of word synonyms, we will here learn models that are targeted at identifying the synonymy relation. Moreover, it is straightforward to learn specialized models using our framework for different word types, such as *nouns*, *verbs* and *adjectives*. Our results suggest that learning typed word similarity measures in identifying synonymy is indeed advantageous.

Interestingly, the tasks of extracting named entity classes and general word semantic similarity from text corpora have been treated separately in the literature. Indeed, these tasks reflect different phenomena in the word graph. Named entity coordinate terms correspond to a large number of named terms, all of which are instances of a specific semantic class (such as *person*, or *city*). Accordingly, these instances appear in highly regular lexical and syntactic neighborhoods. For example, as discussed before, *person* names often appear with the same modifier (e.g. 'Mr.', 'Dr.'), or with the same verb (e.g. 'said'). Therefore, named entity mentions are often retrieved based on distinctive lexical and syntactic patterns, using bootstrapping

Adjectives	contemporary : modern	infrequent : rare
	immediate : instant	dedicated : committed
	lethal : deadly	necessary : essential
	particular : specific	pressing : urgent
	deliberate : planned	informal : casual
	gay : homosexual	isolated : lonely
	dubious : doubtful	legitimate : valid
	infamous : notorious	constant : fixed
	imperative : vital	exact : precise
	lucid : clear	economic : profitable
	intelligent : clever	essential : fundamental
	prosperous : affluent	attractive : appealing
Nouns	commencement : graduation	murderer : assassin
	convention : conference	disaster : catastrophe
	destiny : fate	discount : reduction
	hunger : starvation	impediment : obstacle
	hypothesis : speculation	homicide : murder
	material : fabric	measure : degree
	movie : film	interplay : interaction
	possibility : opportunity	inflow : influx
	remorse : regret	meeting : assembly
	association : organization	ballot : poll
	comfort : consolation	bid : tender
Verbs	answered : replied	oversee : supervise
	conform : comply	received : got
	disappeared : vanished	admitted : confessed
	cited : quoted	began : started
	diminished : decreased	closes : shuts
	enquire : investigate	confine : restrict
	evaluated : assessed	disclose : reveal
	inspected : examined	illustrate : demonstrate
	renewed : resumed	assure : guarantee
	demonstrated : protested	illuminated : clarified
	responded : replied	nominated : appointed
	renewed : resumed	

Fig. 4. Word synonym pairs: the left words were used as query terms in our experiments.

techniques (Hearst 1992; Collins and Singer 1999). General words, on the other hand, are associated with small synonym sets. In addition, there is higher variability of the lexical and syntactic contexts in which general words appear due to different word selectional preferences (e.g. Thater, F'urstenau and Pinkal 2010). The evaluation of general word similarity is therefore typically addressed using distributional similarity models. In this section, we show that the graph-based corpus representation, using the proposed adaptive graph walk-based inference methods, can be effectively applied to the task of evaluating general word synonymy.

7.1 Dataset and corpus

We have constructed a set of example queries that describe word synonym pairs, distinguishing between nouns (twenty-two examples), adjectives (twenty-four examples) and verbs (twenty-three examples). The synonym pairs considered correspond to popular examples given on websites dedicated to teaching English as a second language. (Since such sources listed only a handful of adverbs, we did not include adverbs in this case study.) The full synonym list that constitutes our dataset is given in Figure 4. Alternatively, one can consider the synonym sets included in a lexico-semantic ontology, such as WordNet, to construct a dataset for learning and evaluation. However, the synonymy distinctions annotated in WordNet are often intricate

and context-dependent. We choose to focus on coarse synonymy in this case study, as we are interested in learning general, rather than word-specific, synonymy relations.

We constructed a corpus of parsed text using the British National Corpus (BNC; Burnard 1995) for the experiments. The full BNC corpus is a 100-million-word collection of samples of written and spoken language from multiple sources, designed to represent a wide cross-section of contemporary British English. We extracted sentences from the BNC corpus that contained synonymous words. The number of extracted sentences was limited to 2000 per word. In addition, for infrequent words, we extracted more example sentences from AP articles included in the AQUAINT corpus (Bilotti *et al.* 2007) so as to increase the number of sentences per word to at least 300, if possible. The constructed corpus, BNC+AP, includes 1.3 million words overall. This corpus was parsed using the Stanford dependency parser. While named entity tags are included in the AP corpus, this information is not readily available for the BNC corpus. The nodes of the constructed graph therefore correspond to individual words in this case. The corresponding graph included about 400 thousand *word mention* nodes, about 40 thousand *term* nodes and 1.7 million edges overall.

7.2 Experiments

In our experimental setting, the query distribution V_q consists of a single term of interest. We assume that the word type of the query term is known. Rather than ranking all terms in response to a query, we use available (noisy) part of speech information to narrow down the search to the terms of the same type as the query term, e.g. for the query 'film' we retrieve nodes of type $\tau = noun$.

In preliminary experiments, we found that while weight tuning improves graph walk performance, the quality of the local graph walk models was relatively poor on the synonym extraction dataset. These results are in line with the findings reported before (Figure 3). An analysis of the word graph showed that out of the synonym word pairs included in our dataset, 12.5 percent, 9.5 percent and 4.5 percent of adjective, verb and noun word pairs, respectively, co-appeared within a single sentence in the corpus. Since the Personalized PageRank graph walk schema is biased toward nodes linked over shorter paths in the graph, this means that most synonyms, connected over relatively long cross-sentence connecting paths, are assigned low probability scores by the graph walk. In addition, applying reranking to the top results of the graph walk models gave low performance in this case. We observed that due to the small number of correct examples available in this dataset (having a single correct answer specified per query, compared with many relevant terms in the named entity extraction case study) on one hand, and the high variability of contexts in which general words appear on the other hand (demoting the importance of lexical path features), this dataset was not large enough to allow effective learning by the discriminative reranking model. Hence, in this case study we focus on the PCW method, which incorporates high-level information about the connecting paths as part of the random graph walk process.

We applied the PCW method, learning separate models for *nouns*, *verbs* and *adjectives*. The path trees were constructed using the paths leading to the node known

to be a correct answer, as well as to the otherwise irrelevant top-ranked ten terms, per query. In the experiments, we considered paths that are six segments long. Such paths represent distributional similarity phenomena, allowing a direct comparison against the DV method. In conducting the constrained walk, we applied a threshold of 0.5 to truncate paths associated with lower probability of reaching a relevant response.

Details about the implementation of the DV method are provided in Section 6. We use here the union of the top 300 terms retrieved by path-constrained walk as candidate terms for dependency vectors. For all queries, these sets of terms included the correct answer annotated. We evaluate the following variants of dependency vectors: having inter-word similarity computed using Lin's measure (DV-Lin; Lin 1998), or using cosine similarity (DV-Cos). In addition, we consider a non-syntactic variant, where a word's vector consists of its co-occurrence counts with other terms (using a window of two words); that is, ignoring the dependency structure (CO-Lin).

One difference between distributional methods and graph walk methods is that distributional similarity measures incorporate heuristics that account for the effect of non-informative word co-occurrences, where co-occurrences of the target word with highly frequent words (Manning and Schütze 1999), or with otherwise highly infrequent words (Hughes and Ramage 2007), are often non-informative. For example, Lin's similarity metric, which was designed for the task of evaluating general word similarity from corpus statistics, is based on the mutual information measure, downweighting frequent word co-occurrences. In order to demote the effect of uninformative word co-occurrences, we evaluate the PCW approach also in settings where random and noisy edges have been eliminated from the underlying graph. Specifically, dependency links in the graph were associated with pointwise mutual information (PMI) scores of the linked word mention pairs (Manning and Schütze 1999); edges with low scores are assumed to represent word co-occurrences of low significance, and so are removed. We empirically set the PMI score threshold to 2.0, using cross validation (PCW-P).[8] Finally, for comparison purposes, in addition to the specialized PCW models, we also learned a uniform model over all word types in these settings; that is, this model is trained using the union of all training examples, being learned and tested using a mixture of queries of all types (PCW-P-U).

7.3 Results

Considering the limited number of positive examples available in our dataset, we use cross validation rather than a fixed split into train and test sets in the evaluation. Table 2 gives the results of 10-fold cross-validation experiments in terms of MAP. Note that for the purpose of evaluation, we discard terms in the ranked list, which are known inflections of the query or target terms (e.g. for the word pair 'nominated-appointed', the terms 'nominate', 'nominates', 'appoint' and 'appoints' were eliminated from the output ranked lists.)

As shown in Table 2, the DV model applied using Lin similarity (DV-Lin) performs best among the vector-based models. The improvement achieved due to edge

[8] Eliminating low PMI co-occurrences has been shown to be beneficial in modeling lexical selectional preferences recently, using a similar threshold value (Thater *et al.* 2010).

Table 2. *10-fold cross-validation results: MAP*

	Nouns	Verbs	Adjectives	All
CO-Lin	0.34	0.37	0.37	0.37
DV-Cos	0.24	0.36	0.26	0.29
DV-Lin	0.45	0.49	0.54	0.50
PCW	0.47	0.55	0.47	0.49
PCW-P	**0.53**	**0.68**	**0.55**	**0.59**
PCW-P-U	0.49	0.65	0.50	0.54

weighting compared with the co-occurrence model (CO-Lin) is large, demonstrating that syntactic structure is very informative for modeling word semantics (Padó and Lapata 2007). Interestingly, the impact of applying the Lin similarity measure versus cosine (DV-Cos) is even more profound. Unlike the cosine measure, Lin's metric allows one to downweight uninformative word co-occurrences.

Among the PCW variants, the specialized PCW models achieve performance that is comparable with the state-of-the-art DV measure (DV-Lin). Further, removing uninformative word co-occurrences from the graph (PCW-P) leads to further improvements, yielding the best results over all word types. Finally, the graph walk model that was trained uniformly for all word types (PCW-P-U) outperforms DV-Lin, showing the advantage of *learning* meaningful paths. Notably, the uniformly trained model is inferior to PCW trained separately per word type in the same settings (PCW-P). This suggests that learning *specialized* word similarity metrics is beneficial.

Figure 5 gives a more detailed view of these results. The figure shows the cumulative recall at the top ranks of the retrieved lists, down to rank 40, for the best performing DV-Lin and PCW-P methods. The figure shows that the performance of PCW-P dominates that of DV-Lin in general, and at the top ranks in particular.

Figure 6 provides a view of the top-ranked *terms* output by the DV-Lin and PCW-P methods for a couple of examples, corresponding to the queries $V_q = \{$'isolated'$\}$, retrieving terms known to be *adjectives*, and $V_q = \{$'oversee'$\}$, retrieving *verb* terms. Along with the ranked lists, the figure details the scores assigned by the methods to each term. As shown, given the query term 'isolated', PCW-P ranks the synonym 'lonely' at the first rank (in bold); the DV-Lin method ranks this term at rank 41 (not shown). There are multiple terms, however, that appear at the top ranks of both lists. Specifically, the terms 'small' and 'rural' are included in both lists; these terms are semantically related to the query term, as rural, or small, places are often isolated. In addition, the terms 'affluent' and 'particular' were ranked highly by both methods; we found that these terms, similar to the query term, are often used to describe 'areas', 'towns', 'groups' and so forth. (The term 'particular' and the query term 'isolated' also appear in the neighborhood of terms such as 'cases', or 'issues'.) Using PCW-P, the term 'lonely' is assigned a high weight in response to the query 'isolated' due to common adjectival complement relations with the verb 'feel'; in addition, 'lonely' is found to be related to the 'isolated' based on conjunctions of both terms with adjectives such as 'frustrated' and 'depressed'. Both the complement and conjunction relations are not assigned a high score based on the linguistically

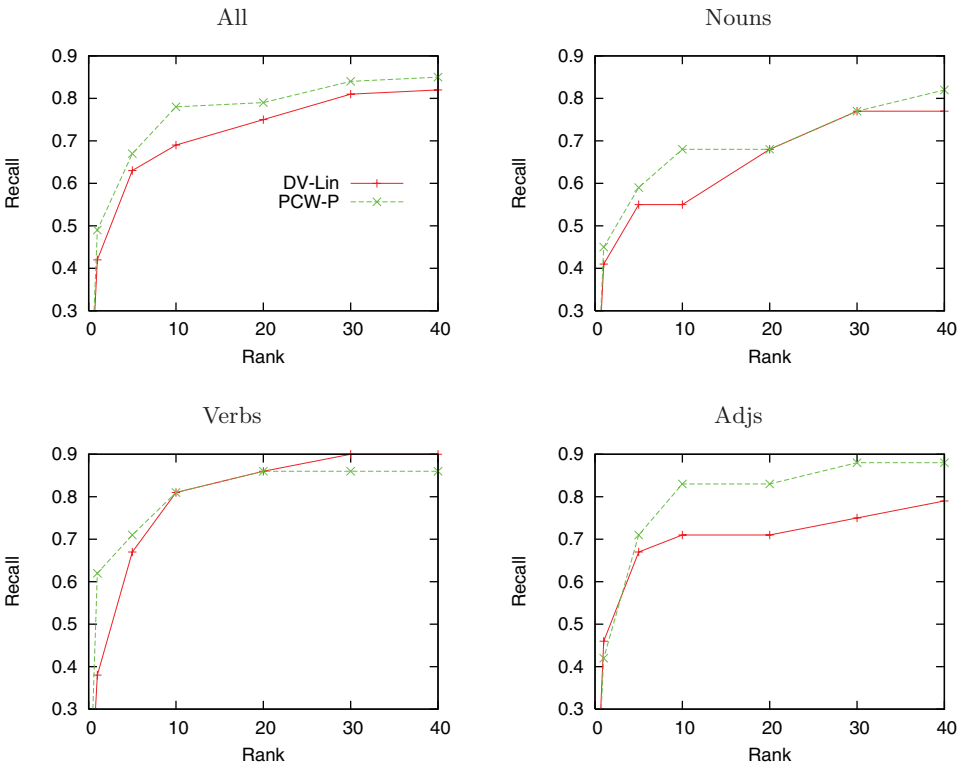

Fig. 5. (Colour online) Synonym extraction results: cumulative recall at top ranks.

	V_q ={'isolated'}				V_q ={'oversee'}			
Rank	DV-Lin		PCW-P		DV-Lin		PCW-P	
1	affluent	0.004	**lonely**	0.015	**supervise**	0.004	**supervise**	0.047
2	particular	0.003	different	0.011	operates	0.003	approaches	0.012
3	established	0.003	held	0.010	planning	0.003	case	0.012
4	limited	0.003	affluent	0.010	established	0.003	concerns	0.011
5	class	0.003	prosperous	0.009	planned	0.003	bills	0.010
6	prosperous	0.003	small	0.009	appointed	0.003	sustain	0.011
7	small	0.003	rural	0.009	involved	0.003	levels	0.010
8	rural	0.003	local	0.007	require	0.003	firms	0.010
9	coming	0.002	clear	0.006	resume	0.003	campaign	0.010
10	major	0.002	commanding	0.006	committed	0.003	approve	0.009
11	communist	0.002	specific	0.006	make	0.003	restrict	0.009
12	developed	0.002	particular	0.006	confine	0.003	work	0.008

Fig. 6. Synonym extraction results: top-ranked *term* lists and the associated scores output by the DV-Lin and PCW-P methods for a couple of example queries.

motivated weighting scheme used by DV-Lin. In the second example given, both methods rank the relevant synonym 'supervise' at the top rank in response to the query 'oversee'. Both terms are often linked over a *direct object* relation with terms such as 'work', 'repairs', 'production', as well as share a *subject* relation with terms such as 'board' or 'director'. The *subject* and *object* relations are weighted highly by both methods. In addition, PCW-P models clausal complements relations (*xcomp*)

with verbs such as 'employed' or 'appointed' (as in 'employed to oversee/supervise'), and conjunction relations with other verbs such as 'regulate' and 'ensure'. Note that PCW-P assigns a significantly higher score to the correct answer compared with the following candidates, whereas DV-Lin ranks other related terms in closer proximity to this synonym. In summary, there are several reasons why the PCW method is advantageous on these examples. While DV-Lin is designed to capture a general semantic similarity notion, PCW-P (as well as the other learned PCW variants) is targeted at finding a specific flavor of similarity, namely *synonymy* in this case study. Thus, the relative importance assigned to different edge types is adapted to the task at hand. In addition, unlike distributional similarity methods, which aggregate lexico-syntactic evidence at sentence level, PCW models information about the full paths connecting the query and target terms, allowing one to capture symmetry and other high-level properties of the common neighborhoods modeled. Finally, we find that the automatically tuned PCW-P method models richer syntactic paths than the manually tuned path set used by DV-Lin.

In our experiments, prominent paths that were found to provide strong positive evidence for noun synonym extraction included:

$$\ldots \xrightarrow{nn} \ldots \xrightarrow{nn-inv} \ldots$$
$$\ldots \xrightarrow{nn} \ldots \xrightarrow{amod-inv} \ldots$$
$$\ldots \xrightarrow{dobj-inv} \ldots \xrightarrow{dobj} \ldots$$
$$\ldots \xrightarrow{amod} \ldots \xrightarrow{amod-inv} \ldots$$
$$\ldots \xrightarrow{conj\_and-inv} \ldots \xrightarrow{conj\_and} \ldots$$
$$\ldots \xrightarrow{prep\_of-inv} \ldots \xrightarrow{prep\_of} \ldots$$

As one may expect, the salient paths in the models learned are mostly symmetrical. There are, however, multiple positively weighted non-symmetrical paths learned by the model. The path '$\ldots \xrightarrow{nn} \ldots \xrightarrow{amod-inv} \ldots$' included in the list above reflects a frequent parsing error, where modification is confused with noun-compound relationship; e.g. the expression 'natural disaster' was erroneously parsed as a noun-compound in our corpus, where the relation between the word pair 'natural catastrophe' was correctly parsed as modification. Another path, '$\ldots \xrightarrow{nn-inv} \ldots \xrightarrow{prep\_of-inv} \ldots$', reflects a valid syntactic variation; e.g. 'disaster management' (a noun-compound) versus 'management of catastrophe'.

In comparison, the set of weighted paths learned per the task of verb synonym extraction was different, including the following example of prominent paths:

$$\ldots \xrightarrow{dobj} \ldots \xrightarrow{dobj-inv} \ldots$$
$$\ldots \xrightarrow{nsubj} \ldots \xrightarrow{nsubj-inv} \ldots$$
$$\ldots \xrightarrow{xcomp-inv} \ldots \xrightarrow{xcomp} \ldots$$
$$\ldots \xrightarrow{advcl-inv} \ldots \xrightarrow{advcl} \ldots$$
$$\ldots \xrightarrow{infmod-inv} \ldots \xrightarrow{infmod} \ldots$$

Finally, examples of significant paths learned in the adjective synonym extraction models include:

$$\ldots \xrightarrow{amod-inv} \ldots \xrightarrow{amod} \ldots$$
$$\ldots \xrightarrow{advmod} \ldots \xrightarrow{advmod-inv} \ldots$$
$$\ldots \xrightarrow{nn-inv} \ldots \xrightarrow{amod} \ldots$$
$$\ldots \xrightarrow{conj\_and-inv} \ldots \xrightarrow{conj\_or} \ldots$$

These different sets of paths demonstrate the utility of learning separate models for specialized notions of word similarity. While many of the syntactic relations included in the top paths learned correspond to linguistically salient relations, such as nominal subject (nsubj) and direct object (dobj), we found that a conjunction relation of nouns and verbs with the same context word provided good supportive evidence of word synonymy; common preposition arguments were found to be meaningful as well in inferring noun synonymy, and so forth. Importantly, the models learned reflect phenomena observed in the underlying (noisy) parsed texts, and may be adapted to other text genres, using different parsing conventions.

8 Design and scalability considerations

In this section, we are interested in assessing the effect of the framework's design choices on performance. We first evaluate graph walk performance, varying the graph walk length and reset probability. In addition, we examine the effect of pruning the path-constrained walk on performance and scalability. We provide empirical evidence using the named entity coordinate extraction corpora and tasks in this section, but similar trends have been observed for the general word synonym extraction case study. Further discussion of design choices concerning the weight tuning and reranking learning methods, as well as their combination, is available elsewhere (Minkov and Cohen 2010).

8.1 Graph walk

The graph walk framework includes two parameters: the reset probability γ; and, as we perform *finite* graph walks, the length of the walk K. In this section we discuss the effect of these parameters on performance.

In all of the experiments reported thus far, the reset probability was set to $\gamma = 0.5$. In experiments conducted on the tasks and corpora evaluated in this work, we found that the value of γ had negligible effect on the actual produced rankings. These results are in line with previous findings, showing that while the reset probability parameter affects the actual scores assigned to the graph nodes, it does not change their ranking (Page *et al.* 1998; Agirre and Soroa 2009; Minkov and Cohen 2010).

We further report empirical results of tuning the graph walk length. Figure 7 shows performance for the city name (left) and person name (right) extraction tasks varying the walk length K. The graphs report results for the MUC corpus, which is fully annotated, thus allowing us to show performance in terms of a precision-recall curve. Both graphs in Figure 7 demonstrate clearly that increasing the graph walk

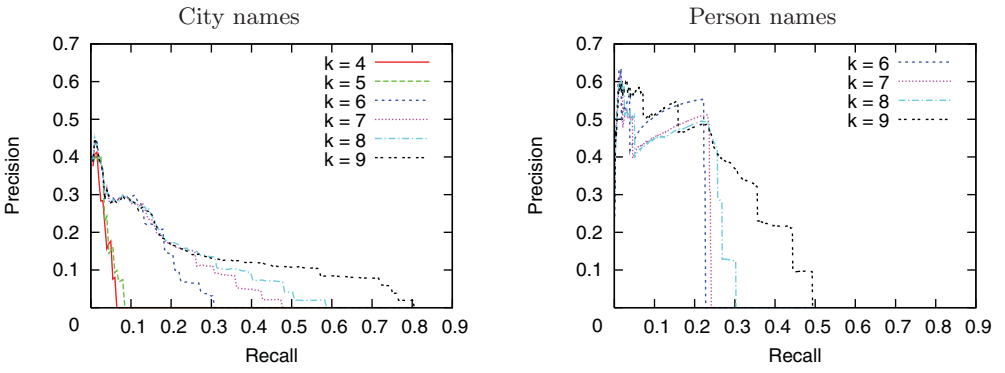

Fig. 7. (Colour online) Precision-recall curves varying the walk length K for city name extraction (top) and person name extraction (bottom). The left graphs show full curves, and the right graphs focus on the top of the lists (down to recall 0.2). These results were all generated using the MUC corpus.

length improves recall. For both tasks, short walks where $k \leqq 4$ yielded poor recall. (For the person name extraction task, recall was near zero, and the corresponding curves were eliminated from the figure.) The reason for this low recall is that there are relatively few relevant nodes that can be reached over short connecting paths in this domain. For example, the two-hops path 'mention – conj-and – isa' models a conjunction relation between words appearing in the same sentence. This type of evidence is relatively scarce in our experimental corpora, occurring more frequently for city names than for person names. (Co-occurrence frequency within the same sentence is generally low for word synonyms, or hypernyms; Snow *et al.* 2005.) The majority of meaningful paths are of length six in the graph; e.g. the path that models a common direct object or subject arguments is of length six. Increasing the walk beyond length $K = 6$ in this domain improves recall, as shown in the figure; however, the additional nodes reached in longer walks are generally added at the bottom of the retrieved list due to the exponential decay associated with the walk. Based on these results, we set the walk length in the experiments to $K = 6$.

8.2 Path-constrained walks

The number of distinct paths observed in our experiments, using constrained graph walks up to length $K = 6$, ranged roughly from 400 to 1500. The corresponding path trees consist of roughly 500 to 3500 nodes respectively. In general, the number of distinct paths observed increases with the number of examples, as well as with the size of the corpus. In the experiments conducted, highly meaningful paths (reported in Sections 6.3 and 7.3) were found to be highly frequent, and were observed in most of the examples used. In general, pruning infrequent paths from the path tree may be desired to avoid over fitting and for efficiency considerations. It is possible, however, that infrequent paths be useful in some domains; we recommend tuning the training set size empirically by means such as cross validation.

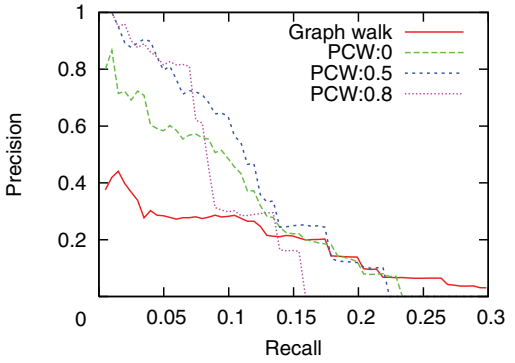

Fig. 8. (Colour online) Precison-recall performance for city name extraction from the MUC corpus for path-constrained walks with varying thresholds, and graph walks with uniform weights.

The PCW algorithm allows the elimination of paths associated with a low probability of reaching relevant target nodes. Specifically, it is straightforward to prune paths from the path tree for which the estimated path probability is below a pre-set threshold. Next we discuss and empirically evaluate the effect of pruning the path-constrained walks on performance and scalability.

Figure 8 shows the results of applying the PCW variant to the task of city name extraction and the MUC corpus using various thresholds. Specifically, we have evaluated thresholds of value 0 (PCW:0), considering all the paths in the corresponding path tree; 0.5 (PCW:0.5), following paths that lead to a majority of relevant nodes only and 0.8 (PCW:0.8), following paths that lead to a strong majority of relevant nodes. Results are shown in terms of a precision-recall curve, using the MUC corpus gold labels. In addition to PCW, the figure includes the performance of graph walks with uniform weights for reference. It is shown that applying a higher threshold to the path-constrained walk leads to improved precision, in this case, at the top ranks. It is also shown, on the other hand, that using a higher threshold yields lower overall recall as the result of decreased path coverage. Based on these results, we have elected to apply a threshold of 0.5 in our experiments, as this threshold gives good performance both in terms of precision and recall. In general, the utility of a threshold of a particular value may be task- or domain-dependent, and can be tuned as a system parameter. In addition, one should consider the effect of the selected threshold on scalability.

In order to discuss the effect of the path-constrained walk on processing time and memory requirements, we first give details on our implementation of the graph walk, including empirical query processing times per our experimental corpora. The reported results in this paper were obtained using a commodity PC with 4-GB RAM, where graph information has been loaded to memory. The option of online computation may be less desirable given larger graphs due to longer response times and memory requirements. To evaluate performance on large graphs, we constructed three intermediate size corpora that include MUC, and incrementally larger portions of the AP corpus. Specifically, the intermediately sized corpora include a quarter of

Table 3. *Node and edge counts of incrementally larger corpora constructed for graph walk scalability assessment*

Corpus	Nodes (K)	Edges (K)
MUC	82	244
MUC+1/4AP	326	1077
MUC+1/2AP	564	1910
MUC+3/4AP	785	2682
MUC+AP	1030	3550

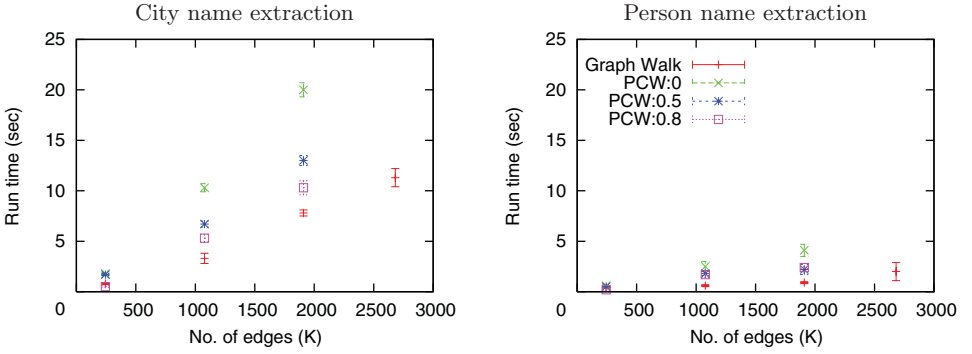

Fig. 9. (Colour online) Average query processing time and standard deviation (sec) for the named entity coordinate extraction tasks, using graph walk of $k = 6$ steps and path-constrained graph walk with varying thresholds.

the AP corpus (MUC+1/4AP); about a half of the AP corpus (MUC+1/2AP) and about three quarters of the AP corpus (MUC+3/4AP). The number of nodes and edges of each corpus are detailed in Table 3.

In the experiments, we observed the processing time per query, t_i, averaged over the queries in the test set of each dataset. We obtained five such observations in repeated runs, for which we report the average: $\sum_{i=1}^{5} t_i/5$. Using the MUC corpus, query processing time was 0.4 seconds on average for the person name extraction task and 0.8 seconds on average for the city name extraction task. The difference in processing times between the two tasks is due to a larger number of city name occurrences in the corpus. (Most of the person names included in the experimental datasets correspond to only few mentions in the corpus such that a smaller number of edges is traversed.)

Figure 9 shows the average query processing times for MUC, as well as for the larger corpora, applying graph walks. Results are displayed as a function of corpus size (in terms of number of edges; see Table 3). As shown, average query processing time gets substantially longer, up to 11.3 seconds on average per query for the MUC+3/4AP corpus and the city name extraction task. In general, the implementation scheme applied in our experiments can be improved by using better machinery as well as by distributed computing. We therefore expect processing times

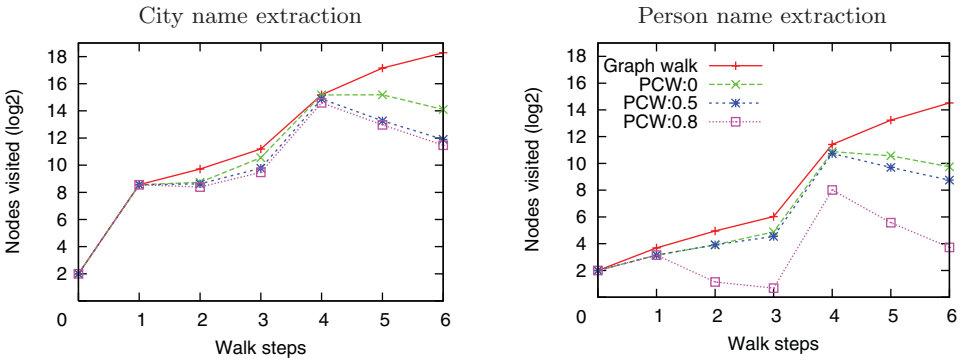

Fig. 10. (Colour online) The cumulative number of nodes visited at each step of the graph walk using the MUC+AP corpus, for city name extraction and person name extraction, applying unconstrained graph walk and path-constrained walk with varying thresholds.

to be shorter using optimized systems, as well as using algorithms that approximate the graph walk (e.g. Fogaras *et al.* 2005).

Figure 9 also shows the empirical average query processing time, applying the path-constrained walk with no threshold (PCW:0); with a threshold of 0.5 (PCW:0.5) and with a high threshold (PCW:0.8) on these corpora. (Results of the path-constrained walk for the MUC+3/4AP are not reported due to high memory requirements, as discussed later in this section.) The results indicate that longer processing times are required using the path-constrained walks, compared with the unconstrained graph walk. As expected, the processing times shorten as the threshold applied increases.

Rather than process the graph walk using the machine's memory, it is also possible to store the graphs in secondary memory. We used the open-source database package Sleepycat to store the user-defined nodes and edges. This allowed us to execute the graph walk for the large MUC+AP corpus (Table 1). The cumulative number of graph nodes visited at each step of the graph walk for the MUC+AP corpus is presented in Figure 10 (logarithm scale). It is shown in the figure that the number of nodes visited at each step of the walk starts dropping at $k = 4$ using the path-constrained walks. Constraining the graph walk to follow a path tree reduces the number of nodes (and edges) traversed in the walk for a couple of reasons. First, the path tree only represents paths observed within the set of connecting paths leading to the top nodes ranked in training examples, where other possible paths are assumed as irrelevant and discarded from the path-constrained walk. In addition, increasing the threshold on the probabilities associated with the path tree edges eliminates more (possibly frequent) paths.

While the path-constrained walk limits the number of nodes (and edges) that are traversed in the walk, it requires the processing of all combinations of a node and its unique histories. In the experiments reported in Figure 9, these added processing requirements overcome the savings due to node pruning in terms of running time. (See the processing times of the base graph walk in the figure as reference.) In addition, maintaining graph and path tree node pairs requires additional memory in comparison to the unconstrained walk. Therefore, we conclude

that path-constrained walks, while contributing to performance, involve overall an additional computational cost. Path-constrained walks are expected to save on processing times, however, in case the graph is accessed from a secondary memory. In that case, node pruning will reduce the cost involved in disk access.

9 Conclusion

We have described a novel but natural representation for a corpus of dependency parsed text as a labeled directed graph. Given the graph, semantic relatedness between word types can be derived using similarity queries in a general purpose graph walk-based query language. In addition, learning techniques can be applied in this framework to adapt the similarity metric to the type of word similarity sought. Available learning methods include tuning of the weights associated with different dependency relations, and candidate reranking using path information and other high-level features derived from the graph walk. In this paper, we have introduced another learning approach, a path-constrained graph walk variant, in which the graph walk is guided by high-level knowledge about meaningful paths learned from training examples. The PCW method outlined is general and can be readily applied to other domains. While it involves an additional computation cost compared with the memory-less random walk, we have shown that this method yields improved performance, and that computation costs can be controlled using pruning.

We have evaluated this framework through a couple of case studies, referring to the tasks of named entity coordinate term extraction and general word synonym extraction from text corpora. We have found that in the studied language domain, learning using high-level information about the paths traversed in the walk was beneficial compared with the local information modeled by the graph walk process. In particular, learning path information allows one to assign higher importance to specific cross-sentence word lexico-semantic neighborhoods. Overall, the proposed framework enables learning of semantic similarity measures based on both within-sentence and cross-sentence (distributional) contextual evidence. Compared with the state-of-the-art distributional syntactic vector space model (dependency vectors), in which fixed weights are set manually per different syntactic paths for context modeling, we learn path weights from examples dynamically, generating a specialized graph walk model for each query type. We have shown that this results in improved extraction performance in many cases for both tasks evaluated using small to medium corpora. We have further shown that learning specialized similarity models at both task and word type granularity can improve the performance of the similarity metric.

In the future, we would like to apply this framework to the extraction of additional specialized word similarity flavors, e.g. hyponymy. While within-sentence path learning has been applied toward extracting hyponym pairs, it has been indicated that within-sentence co-occurrence of hyponyms was rare (Snow *et al.* 2005). We believe that graph walk across sentences can capture more hyponym pairs from a given corpus. Further, the outlined framework can be applied to the extraction of more specialized notions of word relatedness, as in relation extraction (Culotta

and Sorensen 2004; Bunescu and Mooney 2005). Another venue of future work is enriching the modular graph representation with available ontologies, adding known semantic (and morphological) relations between words. We believe that a graph walk similarity metric inferred from such a heterogeneous graph, which integrates various resources and includes a large variety of relations, should benefit from learning, considering high-level phenomena in the graph. Finally, we are interested in exploring more scalable graph walk algorithms and learning methods for inference using the word graph (Lao and Cohen 2010).

Acknowledgments

This work was supported by the grant from the United States-Israel Binational Science Foundation, No. 2010090.

References

Agarwal, A., Chakrabarti, S., and Aggarwal, S. 2006. Learning to rank networked entities. In *The Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, August 20–23, Philadelphia, USA.

Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Pasca, M., and Soroa, A. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *HLT-NAACL*.

Agirre, E., and Soroa, A. 2009. Personalizing pagerank for word sense disambiguation. In *Proceedings of the North American Chapter of the Accosiction of Computational Linguistics (NAACL-HLT)*, May 31-June 5, Boulder, Colorado.

Barzilay, R., and Elhadad, M. 1999. Text summarizations with lexical chains. In I. Mani and M. Maybury (eds.), *Advances in Automatic Text Summarization*, pp. 111–129. Cambridge, MA: MIT.

Bilotti, M. W., Ogilvie, P., Callan, J., and Nyberg, E. 2007. Structured retrieval for question answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Rearch & Development on Information Retrieval*, July 23–27, Amsterdam, The Netherlands.

Bunescu, R. C., and Mooney, R. J. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and Conference of Empirical Methods in Natural Language Processing (HLT/EMLNP)*, October 6–8, Vancouver, B.C., Canada.

Burnard, L. 1995. *Users Guide for the British National Corpus*. British National Corpus Consortium. Oxford, UK: Oxford University Computing Service.

Cohen, W. W., and Minkov, E. 2006. A graph-search framework for associating gene identifiers with documents. *BMC Bioinformatics* **7**(440)

Collins, M. 2002. Ranking algorithms for named-entity extraction: boosting and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, July 6–12, Philadelphia, PA, USA.

Collins, M., and Koo, T. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics* **31**(1): 25–69.

Collins, M., and Singer, Y. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, June 21–22, University of Maryland, MD, USA.

Collins-Thompson, K., and Callan, J. 2005. Query expansion using random walk models. In *Proceedings of the ACM 14th Conference on Information and Knowledge Management (CIKM)*, October 31–November 5, Bremen, Germany.

Culotta, A., and Sorensen, J. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the Joint 42nd Annual Meeting of the Association for Computational Linguistics and the Conference on Empirical Methods in Natural Language Processing (ACL-EMNLP)*, July 21–26, Barcelona, Spain.

de Marneffe, M.-C., MacCartney, B., and Manning, C. D. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, May 24–26, Genoa, Italy.

Diligenti, M., Gori, M., and Maggini, M. 2005. Learning web page scores by error back-propagation. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, July 30–August 5, Edinburgh, Scotland.

Erkan, G., and Radev, D. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)* **22**: 457–479.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.

Fogaras, D., Rácz, B., Csalogány, K., and Sarlós, T. 2005. Towards scaling fully personalized pagerank: algorithms, lower bounds, and experiments. *Internet Mathematics* **2**(3): 333–358.

Grefenstette, G. 1994. *Explorations in Automatic Thesaurus Discovery*. Dordrecht, Netherland: Kluwer.

Harrington, B. 2010. A semantic network approach to measuring relatedness. In the *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, August 23–27, Beijing, China.

Hassan, A., and Radev, D. 2010. Identifying text polarity using random walks. In *The 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, July 11–16, Uppsala, Sweden.

Haveliwala, T. H. 2002. Topic-sensitive PageRank. In *Proceedings of the Eleventh International World Wide Web Conference (WWW)*, May 7–11, Honolulu, Hawaii, USA.

Hearst, M. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings o thef 14th International Conference on Computational Linguistics (COLING)*, August 23–28, 1992, Nantes, France.

Hughes, T., and Ramage, D. 2007. Lexical semantic relatedness with random graph walks. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, June 28–30, Prague, Czech Republic.

Kamps, J., Marx, M., Mokken, R. J., and de Rijke, M. 2002. Words with attitude. In the *Proceedings of the International Conference on Global WordNet*, January 21–25, Mysore, India.

Keenan, E., and Comrie, B. 1977. Noun phrase accessibility and universal grammar. *Linguistic Inquiry* **8**(1): 63–99.

Koren, Y., North, S. C., and Volinsky, C. 2006. Measuring and extracting proximity in networks. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, August 20–23, Philadelphia, PA, USA.

Lao, N., and Cohen, W. W. 2010. Fast query execution for retrieval models based on path constrained random walks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, July 25–28, Washington, DC, USA.

Lao, N., Subramanya, A., Pereira, F., and Cohen, W. W. 2012. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, July 12–14, Jeju Island, Korea.

Lin, D. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, August 10–14, Université de Montréal, Montréal, Quebec, Canada.

Lin, D., and Pantel, P. 2001. Discovery of inference rules for question answering. *Natural Language Engineering* **7**(4): 343–360.

Manning, C., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.

Mihalcea, R. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of the Conference on Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP)*, October 6–8, Vancouver, British Columbia, Canada.

Mihalcea, R., and Tarau, P. 2004. Textrank: bringing order into texts. In *Proceedings of the Joint 42nd Annual Meeting of the Association for Computational Linguistics and the Conference on Empirical Methods in Natural Language Processing (ACL-EMNLP)*, July 21–26, Barcelona, Spain.

Minkov, E., and Cohen, W. W. 2010. Improving graph-walk-based similarity with reranking: case studies for personal information management. *Transactions on Information Systems (TOIS)* **29**(1): 41–52.

Mirkin, S., Dagan, I., and Geffet, M. 2006. Integrating pattern-based and distributional similarity methods for lexical entailment acquisition. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, July 17–21, Sydney, Australia.

MUC6. 1995. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. Columbia, MD: Morgan Kaufmann.

Navigli, R., and Lapata, M. 2007. Graph connectivity measures for unsupervised word sense disambiguation. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, January 6–12, Hyderabad, India.

Navigli, R., and Lapata, M. 2010. An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**(4): 678–692.

Padó, S., and Lapata, M. 2007. Dependency-based construction of semantic space models. *Computational Linguistics* **33**(2).

Page, L., Brin, S., Motwani, R., and Winograd, T. 1998. The pagerank citation ranking: bringing order to the web. Technical Report, Computer Science department, Stanford University. Working Paper 1999–0120.

Resnik, P., and Diab, M. 2000. Measuring verb similarity. In *The 22nd Annual Conference of the Cognitive Science Society (CogSci)*, Philadelphia, PA.

Roark, B., and Charniak, E. 1998. Noun phrase co-occurrence statistics for semi-automatic lexicon construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, August 10–14, Université de Montréal, Montréal, Quebec, Canada.

Shen, L., and Joshi, A. K. 2005. Ranking and reranking with perceptron. *Machine Learning* **60**(1–3): 73–96.

Snow, R., Jurafsky, D., and Ng, A. Y. 2005. Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of the Nineteenth Annual Conference on Neural Information Processing Systems (NIPS)*, December 5–8, Vancouver, British Columbia, Canada.

Terra, E., and Clarke, C. L. A. 2003. Frequency estimates for statistical word similarity measures. In *Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics (NAACL)*, June 3–8, Montréal, Canada.

Thater, S., F̈urstenau, H., and Pinkal, M. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, July 11–16, Uppsala, Sweden.

Toutanova, K., Manning, C. D., and Ng, A. Y. 2004. Learning random walk models for inducing word dependency distributions. In *Proceedings of the Twenty-first International Conference (ICMl)*, July 4–8, Banff, Alberta, Canada.

van der Plas, L., and Tiedemann, J. 2006. Finding synonyms using automatic word alignment and measures of distributional similarity. In *Proceedings of the 21st International Conference*

on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL), July 17–21, Sydney, Australia.

Wang, R. C., and Cohen, W. W. 2007. Language-independent set expansion of named entities using the web. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007)*, October 28–31, 2007, Omaha, Nebraska, USA.

Wojtinnek, P.-R., Völker, J., and Pulman, S. 2012. Building semantic networks from plain text and Wikipedia with application to semantic relatedness and noun compound paraphrasing. *International Journal of Semantic Computing (IJSC)* (Special Issue on Semantic Knowledge Representation), Vol. 6, No. 1, pp. 67–92.