Walk-based Computation of Contextual Word Similarity

Kazuo Hara[1] *Ikumi Suzuki*[1] *Masashi Shimbo*[2] *Yuji Matsumoto*[2]
(1) National Institute of Genetics, Mishima, Shizuoka, Japan
(2) Nara Institute of Science and Technology, Ikoma, Nara, Japan
kazuo.hara@gmail.com, suzuki.ikumi@gmail.com, shimbo@is.naist.jp,
matsu@is.naist.jp

ABSTRACT
We propose a new measure of semantic similarity between words in context, which exploits the syntactic/semantic structure of the context surrounding each target word. For a given pair of target words and their sentential contexts, labeled directed graphs are made from the output of a semantic parser on these sentences. Nodes in these graphs represent words in the sentences, and labeled edges represent syntactic/semantic relations between them. The similarity between the target words is then computed as the sum of the similarity of walks starting from the target words (nodes) in the two graphs. The proposed measure is tested on word sense disambiguation and paraphrase ranking tasks, and the results are promising: The proposed measure outperforms existing methods which completely ignore or do not fully exploit syntactic/semantic structural co-occurrences between a target word and its neighbors.

KEYWORDS: contextual word similarity, word sense disambiguation, paraphrase, kernel method.

Proceedings of COLING 2012: Technical Papers, pages 1081–1096,
COLING 2012, Mumbai, December 2012.

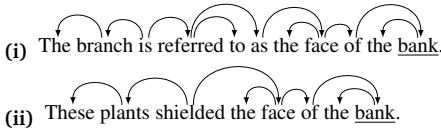
1 Introduction

Word ambiguity poses a great difficulty in natural language processing (Deerwester et al., 1990; Berger and Lafferty, 1999; Navigli, 2009). In document classification, for example, polysemous words may provide spurious evidence for misclassifying documents of different categories into a single category. Synonyms may also cause problems; without the knowledge of two words being synonyms, classifiers might be unable to detect similarity of documents in which they appear.

These problems can be alleviated with the help of contextual similarity (Jurafsky and Martin, 2008, Section 20.7). Polysemous words have identical surface forms, but the contexts in which they appear are often distinct; and even though synonyms have different surface forms, they are expected to appear in similar context.

There has been a volume of work investigating semantic similarity of words in context. However, syntactic or semantic *structure* in the context has not been fully explored. The methods representing context as a *bag of words* or a *bag of n-grams* (Schütze, 1998; Reisinger and Mooney, 2010; Erk and Padó, 2010; Dinu and Lapata, 2010; Giuliano et al., 2009) ignore the underlying syntactic/semantic structure. Recent studies on compositional semantics of words (Mitchell and Lapata, 2008; Erk and Padó, 2008; Thater et al., 2010) take advantage of structure in context, but they only use information of words that directly stand in a predicate-argument relation with the target word.

Consider the following sentences (i) and (ii), shown with the word dependency relations.



In sentence (i), the noun (homonym) *bank* means *"a financial institution"* whereas the one in sentence (ii) means *"the slope of land adjoining a river."*

If these sentences are treated as bags of words, we see that the words unique to (i) and (ii) are respectively {*branch*, *is*, *refer*} and {*plant*, *shield*}, excluding functional words. However, these sets of words are not likely to be sufficient to discriminate the sense of the two *bank*s. The bag-of-*n*-grams representation (Giuliano et al., 2009) collects *n*-grams containing a target word from the context. This representation is still short on distinguishing the senses of the two *bank*s above, as the *n*-grams around them are completely identical up to $n = 5$; i.e., the 5-grams surrounding *bank* are *"the face of the bank"* in both sentences. Recent methods (Mitchell and Lapata, 2008; Erk and Padó, 2008; Thater et al., 2010) exploit direct syntactic/semantic relations with target words, but these methods would still fail in the above examples, because the direct structural neighbors of *bank*s are the same in the two contexts.

On the other hand, syntactic/semantic structural co-occurrences of multiple words, obtained from the dependency relations, seem effective for this example. Consider structural collocations *referred* → *as* → *face* → *of* → *bank* and *shielded* → *face* → *of* → *bank*, which can be obtained if we follow multiple dependency arrows in the dependency structure. The problem is how to compute such co-occurrences efficiently from given syntactic/semantic structures.

Contributions. In this paper, to fully exploit syntactic/semantic co-occurrences of words in contexts, we represent a target word in a context as a *bag of walks* on the *parse graph* (Section 3). In this graph, nodes represent words, and edges represent syntactic/semantic relations between nodes. Such a graph can be readily obtained from the output of state-of-the-art semantic parsers.

We further define the *bag-of-walks* kernel between graph nodes (Section 4). With this kernel, word similarity is calculated as the sum of the similarity of walks starting from target words in their respective parse graphs. Here, the similarity of two walks is the product of the similarity of individual nodes and edges visited during the walks. Since nodes represent words and edges represent syntactic/semantic relations between them, this similarity computation takes into account syntactic/semantic structural co-occurrence of multiple words in context.

We verify effectiveness of the proposed method in two experiments: one on word sense disambiguation and the other on paraphrase ranking (Section 5).

2 Related work

The simplest representation of a context is to regard it as a *bag of words*, or a vector holding the frequency of words co-occurring with the target word in the given context. We call this vector *local context vector* in this paper. Depending on the task, context can be a fixed number of words around the target word, or the sentence or paragraph in which it appears.

We can regard a local context vector as representing a particular meaning the target word conveys in the given context. However, these vectors may be too sparse to provide enough information, especially when the available context is short. Hence, recent studies have focused on the way to make a richer representation of context utilizing information extracted from external resources, such as corpora.

The majority of research along this line utilizes a context-free representation of a word, called *type vector* (Erk and Padó, 2010). The type vector of a word, in its simplest form, is built by aggregating local context vectors made for individual occurrences of the word in the corpus. Thus, it does not represent a specific context of a word, but is a general representation of a word viewed as a type. Still, type vectors can be used as building blocks of *contextualized* feature vectors (Thater et al., 2010) representing the context around the target word.

Schütze (1998) proposed the *word vectors* as one such contextualized feature vector. The word vector of a target word is the sum of the type vectors of the words appearing in its surrounding context. Because it is an aggregate of dense type vectors of different words, we can expect a word vector to hold information richer than a naive, bag-of-words local context vector. More elaborate contextualization has been proposed recently, which combines (e.g., via component-wise vector multiplication) the type vector of the target word with those of the words occurring in the context (Mitchell and Lapata, 2008) in a compositional way. Erk and Padó (2008) and Thater et al. (2010) took a similar compositional approach, but they made type vectors by taking the semantic structure into account, i.e., by counting frequency of words in the corpus that participate in syntactic or semantic relations with the target word.

Erk and Padó (2010) proposed a corpus-based method that does not rely on type vectors. They first make a bag-of-words local context vector $\mathbf{v}$ for a given target word, in a usual manner. They then collect k instances of the target word in a corpus whose contexts are most similar to the one for the target instance (with respect to a predefined similarity measure of contexts). The

local context vectors for these k instances are then added to the original local context vector $\mathbf{v}$ to make it denser. The idea behind this method is that because these k instances appeared in a context similar to that of the target instance, they are likely to have the same sense.

As we have seen above, even the most recent methods do not use syntactic/semantic structure of context at all, or make only limited use of it; namely, only the words that directly participate in a relation (e.g., predicate-argument) with a target word are taken into account.

3 Bag-of-walks representation of word in context

In this section, we introduce the *bag-of-walks* representation of words in context. This representation encodes syntactic/semantic co-occurrences of words around a target word as a collection of walks in a *parse graph*, a graph that can be obtained from the structure output by syntactic/semantic parsers.

We first describe how to make a parse graph from a parser output, and then define a natural random walk model on this graph. The parse graph and the random walk model will be used to define the "bag-of-walks kernels" in Section 4, which allow us to evaluate the similarity of two contexts represented as bags of walks.

For brevity, we assume below that the context is a sentence containing a target word. Smaller fragments (e.g., phrases) can also be used in place of sentences, as long as the relations between words in a fragment can be obtained. For instance, we could use a subtree (corresponding to a phrase) extracted from the dependency tree of an entire sentence.

3.1 Parse graphs

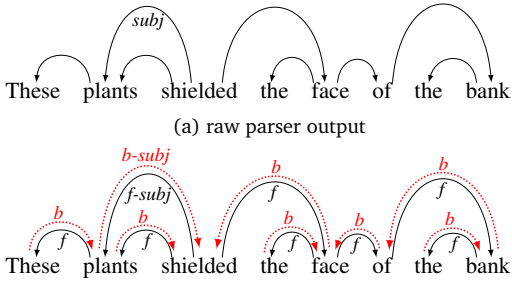
The *parse graph* can be obtained from the output of (semantic) dependency parsers.

Basic dependency parsers output a tree in which nodes are the words in the input sentence and (directed) edges between them represent syntactic head-dependent (dependency) relations. A dependency may be expressed as a directed edge either from the head to the dependent, or from the dependent to the head. The choice is arbitrary, but both head-to-dependent relations and dependent-to-head relations potentially provide useful features for semantic similarity of words. The same should be true for other types of syntactic/semantic relations output by more advanced parsers, such as the *"semantic subject"* of a verb in passive voice. Hence, we represent each individual relation output by the parser as two distinct edges having the same pair of end nodes (words), but with direction opposite to each other. The two edges have the same label representing the type of the relation, but one of them is prefixed *"forward"* (or *"f"* for short) and the other is prefixed *"backward"* (or *"b"*), so that the two edges can be distinguished by the label. Thus for example, if the parser outputs a relation called *"semantic subject"* between two nodes, one edge will be labeled *"forward semantic subject"*, and the other, opposite direction edge will have the label *"backward semantic subject."*

We call a labeled directed graph created in this way a *parse graph*. See Figure 1 for illustration.

3.2 Walks in a graph

We now consider walks in the parse graph starting from the target word. Given a graph G, a walk is a series of nodes and edges, formally defined as follows. Let $\mathrm{Out}(w)$ denote the set of outgoing edges from node w, and $\mathrm{sink}(r)$ denote the sink node of directed edge r. A *walk* $z = (w_0, r_1, w_1, r_2, \ldots, r_t, w_t)$ in graph G is an alternating series of nodes $(w_0, \ldots, w_t)$ and edges



(a) raw parser output

(b) after conversion

Figure 1: Illustration of parse graph construction. (a) The raw dependency structure extracted from the output of the Enju parser. Notice multiple edges from "shielded" to "plants"; the parser outputs a semantic relation named "*subj*" in addition to syntactic dependencies (shown without edge labels). (b) Each relation in the parser output is converted to two edges, to allow "forward" and "backward" (shown as a dotted red arrow) traversal of each relation. In the figure, "*b*" and "*f*" are the shorthands for "backward" and "forward," respectively.

$(r_1, \ldots, r_t)$ in G such that w_{i-1} and w_i are the source and sink of edge r_i; i.e., $r_i \in \mathrm{Out}(w_{i-1})$ and $w_i = \mathrm{sink}(r_i)$ for every $i = 1, \ldots, t$. We call t the *length* of z and denote it by $|z|$. A walk may contain duplicate nodes and edges. Thus it is possible that $w_i = w_j$ or $r_i = r_j$ for some i and j, $j \neq i$.

A walk starting from a target word (node) on a parse graph represents the structural co-occurrence of multiple words around the target word. For example, an entire predicate-argument structure consisting of a predicate p and arguments $\{a_1, a_2, \ldots, a_n\}$ may be represented by a walk $p \to a_1 \to p \to a_2 \to p \to \cdots \to p \to a_n$, traveling between the predicate and each of its arguments alternately; as mentioned above, a walk may visit the same nodes multiple times.

In this way, we represent the context surrounding a target word as a bag of all walks (of bounded length) starting from the target word on a parse graph; we call this a *bag-of-walks* representation of context.

We now introduce a random walk model on a parse graph, as a way to systematically associate a probability (or weight) to each walk in the "bag." In this model, a walk starts from a node corresponding to the target word, and the following process is repeated: (1) Flip a biased coin (with head probability γ). If it comes out heads, terminate the walk. If it comes out tails, proceed to step 2. (2) Choose an edge outgoing from the current node uniformly at random. (3) Follow the edge to the next node. After this process is repeated L times, the walk is terminated, but as step 1 shows, it may also be terminated prematurely at each step with a constant probability γ.

This random walk model assigns probability to each walk of length at most L, starting from the target word in the parse graph. Let w be a node in G, and let $z = (w_0, r_1, w_1, r_2, \ldots, r_{|z|}, w_{|z|})$, be a walk. By the way the random walk model is defined, it is easy to verify that the probability

of taking walk z from node w, written $p(z \mid w)$, is given by

$$p(z \mid w) = \begin{cases} 0, & \text{if } w \neq w_0 \text{ or } |z| > L; \\ \gamma^{I[|z| \neq L]}(1-\gamma)^{|z|} \prod_{i=1}^{|z|} \frac{1}{|\mathrm{Out}(w_{i-1})|}, & \text{otherwise;} \end{cases} \quad (1)$$

where $I[X]$ is an indicator function taking 1 if proposition X holds, or 0 otherwise. Recall that γ is the predefined stopping probability.

As we can see from Eq. (1), walk probability can be tuned by the model parameters L and γ; e.g., the larger the stopping probability γ, the smaller the probability of taking longer walks. These parameters control how much weight should be placed on long-distance dependencies between words, and how far we should look for such dependencies. Section 4 defines the bag-of-walks kernels using the walk probability $p(\cdot)$ as the weight of individual walks.

4 Contextual word similarity via bag-of-walks kernels

This section presents a new method of computing similarity between words occurring in different contexts, which takes syntactic/semantic structural co-occurrences around target words into account. As noted in the previous section, we first build parse graphs for sentences containing target words, using a syntactic or semantic parser. In a parse graph, nodes represent words in the sentence, and edges represent (binary) relations between the words. We then compute the similarity between a pair of target words, represented as nodes in two different parse graphs, using a kernel that defines an inner product between nodes in graphs.

A popular approach to measuring the similarity of structured data is to count shared substructures with the so-called *convolution kernels* (Haussler, 1999; Gärtner, 2003). Our method also takes this approach; to measure the similarity between words in parse graphs, we use a variation of the kernel proposed by Desrosiers and Karypis (2009). This kernel is designed to evaluate the similarity of target nodes in a graph (or possibly in two different graphs) in which nodes and edges are labeled. The basic idea is to compute the number (weight) of identical walks starting from the nodes in respective graphs. Here, two walks are deemed identical if they have the same length and the series of labels along the nodes and edges in the two walks exactly match. We can also relax the requirement of exact matching, and instead allow for varying degree of similarity between different node/edge labels. The resulting kernel computes the node similarity through the degree of similarity of every pairs of (possibly non-identical) walks.

Below, we first give a definition of kernel (or similarity) between two individual walks in the parse graphs (Section 4.1), and then explain how to compute the kernel between two contexts viewed as bags of walks (*bag-of-walks kernel*) in Section 4.2. Under the random walk model presented in Section 3.2, this formulation allows efficient recursive computation without explicitly enumerating all single walks in the graphs.

Let G be a directed graph consisting of a union of two disjoint subgraphs, each of which corresponds to the parse graph for sentences S and S', respectively.

4.1 Walk similarity

We assume two kernel (similarity) functions $K_{\mathrm{word}}(w, w')$ and $K_{\mathrm{rel}}(r, r')$ are at our disposal, which are defined respectively over pairs of nodes (w, w') and pairs of edges (r, r') in G. In parse graphs, nodes represent words, so a natural example for $K_{\mathrm{word}}(w, w')$ is the cosine of

$\mathbf{v}_w$ and $\mathbf{v}_{w'}$, where $\mathbf{v}_w$ is the feature vector (e.g., type vector of Section 2) of word w; for edges, we can simply define $K_{\text{rel}}(r, r') = 1$ if the directed edges r and r' have the same relation label, or 0 otherwise. Using these basic kernels, we define kernel $K_{\text{walk}}(z, z')$ between two walks $z = (w_0, r_1, w_1, r_2, \ldots, r_t, w_t)$ and $z' = (w'_0, r'_1, w'_1, r'_2, \ldots, r'_{t'}, w'_{t'})$. First, if z and z' have different length $t \neq t'$, let $K_{\text{walk}}(z, z') = 0$. For two walks with the same length $t = t'$, we define

$$\begin{aligned} K_{\text{walk}}(z, z') &= K_{\text{word}}(w_0, w'_0) \cdot K_{\text{rel}}(r_1, r'_1) \cdot K_{\text{word}}(w_1, w'_1) \cdot \ \ldots \ \cdot K_{\text{rel}}(r_t, r'_t) \cdot K_{\text{word}}(w_t, w'_t) \\ &= \prod_{i=0}^{t} K_{\text{word}}(w_i, w'_i) \prod_{i=1}^{t} K_{\text{rel}}(r_i, r'_i). \end{aligned} \tag{2}$$

Thus, $K_{\text{walk}}(z, z')$ is the product of the similarity of the pairs of nodes and edges visited along the two walks z and z'.

4.2 Bag-of-walks kernels

Let w and w' be nodes in a graph[1] G. The *bag-of-walks kernels* between w and w', denoted by $K(w, w')$, is defined as a weighted sum of similarity given by K_{walk} between all pairs of walks of length at most L, with each walk starting from w and w', respectively. Here, L is the parameter of the kernel. Formally speaking,

$$K(w, w') = \sum_{\substack{z \in Z(w;L) \\ z' \in Z(w',L)}} p(z \mid w) p(z' \mid w') K_{\text{walk}}(z, z') \tag{3}$$

where $Z(w; L)$ denote the set of all walks of length at most L, starting from w in G. $p(z \mid w)$ is a function that assigns a non-negative weight to walk z. In the rest of the paper, we define $p(z \mid w)$ to be the one given by Eq. (1), i.e., the probability of taking walk z from w according to the random walk model defined in Section 3.2.

4.3 Efficient computation

Naively enumerating all walk pairs (z, z') in Eq. (3) is intractable when L is large, because the number of possible walks grows exponentially with length bound L. However, for the specific weight function $p(z \mid w)$ given by Eq. (1), we can efficiently compute $K(w, w')$ in a recursive manner, as follows.

For $t = 0, 1, \ldots, L$, and any node pair w and w' in G, let $K^{(t)}(w, w')$ be the quantity defined as

$$K^{(t)}(w, w') = \sum_{\substack{z \in Z(w;t) \\ z' \in Z(w';t)}} p(z \mid w) p(z' \mid w') K_{\text{walk}}(z, z'). \tag{4}$$

Eq. (4) has the same form as Eq. (3), except that it is parametrized by the maximum length t of allowed walks. Especially, $K(w, w') = K^{(L)}(w, w')$.

We now show how to compute $K^{(t)}$ recursively over $t = 0, \ldots, L$. For $t = 0$, there exists only one walk pair of length 0, namely, $z^{(0)} = (w)$, $z'^{(0)} = (w')$. Thus we have

$$K^{(0)}(w, w') = K_{\text{walk}}(z^{(0)}, z'^{(0)}) = K_{\text{word}}(w, w'), \tag{5}$$

[1] Note that even if two nodes w and w' appear in different graphs, they can be regarded as nodes in a single graph, namely, the graph union of the two graphs. Hence two nodes can be assumed to belong to a graph G without loss of generality.

where the second equality follows from Eq. (2).

For $t \geq 1$, observe that every walk starting from w will either (i) be terminated immediately with probability γ, or (ii) with probability $(1-\gamma)/|\mathrm{Out}(w)|$, follows an edge $r \in \mathrm{Out}(w)$ and connects to a walk of length at most $(t-1)$ starting from $\mathrm{sink}(r)$, yielding a walk of length at most t as a whole. Recall that $|\mathrm{Out}(w)|$ is the out-degree of nodes w. It follows that

$$K^{(t)}(w,w') = \gamma^2 K_{\mathrm{word}}(w,w') + \frac{(1-\gamma)^2}{|\mathrm{Out}(w)||\mathrm{Out}(w')|} \cdot K_{\mathrm{word}}(w,w') \sum_{\substack{r \in \mathrm{Out}(w) \\ r' \in \mathrm{Out}(w')}} K_{\mathrm{rel}}(r,r') \cdot K^{(t-1)}(\mathrm{sink}(r), \mathrm{sink}(r')). \quad (6)$$

By iteratively computing $K^{(t)}(v,v')$ for all pairs of nodes (v,v') in graph G over $t = 0, 1, \ldots, L$, we eventually obtain the desired quantity $K(w,w') = K^{(L)}(w,w')$, namely, the local structural similarity of nodes w and w' in graph G. The time complexity of this calculation is $O(|G|^2 L)$, where $|G|$ is the number of nodes in graph G. If G comprises two disjoint subgraphs corresponding to the parse graphs for sentence S and S', we can show that the computation time can be reduced to $O(|S||S'|L)$, where $|S|$ and $|S'|$ are the number of words in sentences S and S', respectively; or equivalently, these are the number of nodes in their respective parse graphs. It follows that in practice computation of this bag-of-walks kernel is feasible, because sentences S and S' usually contain only a moderate number of words.

Depending on the structure of the parse graphs and the value of L, we can further save computation. To obtain the value of $K(w_{\mathrm{target}}, w'_{\mathrm{target}}) = K^{(L)}(w_{\mathrm{target}}, w'_{\mathrm{target}})$, we need to compute Eqs. (5) and (6) not for all pairs of words in the sentences, but only for pairs (w,w') such that w and w' are within L steps from the target words w_{target} and w'_{target} in the parse graph.

4.4 Relation to other kernels for structured data

4.4.1 Desrosiers-Karypis kernels

The bag-of-walks kernels are similar to but not identical to the kernels proposed by Desrosiers and Karypis (2009), which also measures the similarity of nodes in the labeled graph using their surrounding structure.

The Desrosiers-Karypis kernels can also be viewed as a special case of Eq (3), but the underlying random walk model $p(\cdot)$ is different. Desrosiers and Karypis use a random walk model that do not cast a bound on the length of walks. In contrast, our model poses a strict upper bound L on the walk length; this model was chosen because co-occurrences comprising hundreds of words are unlikely to be effective for contextual word discrimination. Also, this upper bound reduces computational complexity.

Different random walk models lead to different recursive computations. Observe that the formula for Desrosiers-Karypis kernels (see the unnumbered equation for $\sigma_{u,u'}$ in (Desrosiers and Karypis, 2009, Section 3.3, page 266)) is given as a sum of the probability of generating parallel walks, and every term in the sum is multiplied by a square of stopping probability γ. This is not the case with the bag-of-walks kernels; notice that stopping probability γ is absent from Eq. (5), the base formula for recursive computation.

Another minor difference is that Desrosiers-Karypis kernels do not count the similarity of walks

with zero length. Our kernels take this into consideration, as reflected in Eq. (5) and the first term of Eq. (6).

Both the bag-of-walks kernels and the Desrosiers-Karypis kernels borrow idea from Kashima et al. (2003) who defined a kernel (*marginalized graph kernel*) between two graphs (not graph nodes) on the basis of the marginal probability of parallel random walks taking place in the two graphs. Elegant product graph formulations of various graph kernels can be found in Vishwanathan et al. (2010).

4.4.2 Tree kernels

Tree kernels (Collins and Duffy, 2001; Kashima and Koyanagi, 2002; Moschitti et al., 2008; Croce et al., 2011) efficiently count all common subtrees in two tree-structured data. Although it is tempting to apply these kernels to measure the similarity between the dependency (sub)trees surrounding the target words, it should be noted that our goal is to measure the similarity of particular word pairs in two sentences, and not of the sentences or their entire dependency trees. In particular, tree kernels do not give any special treatment of the target words, and thus they even count subtrees that do not involve a target word at all. Tree kernels are hence not suitable for measuring word similarity. In contrast, the bag-of-walks kernels distinguish target words from the other words in the sentence, as they only count walks starting from the target words. Moreover, tree kernels cannot deal with parser outputs containing semantic relations, as these outputs are in general graphs and not trees.

5 Experiments

In this section, we evaluate the bag-of-walks kernels in two tasks: word sense disambiguation (WSD) and paraphrase ranking. For the WSD task, we use the kernels to construct SVM classifiers. For the paraphrase ranking task in which no training data is provided, we use the kernel value simply as the similarity score used to rank candidate words.

We need to define two basic kernels, K_{word} between words (nodes) and K_{rel} between relations (edges) as the building blocks of the bag-of-walks kernels K (Eq. (3)). Throughout the experiments, we let K_{word} be the cosine between context-independent type vectors for words (see below). For K_{rel}, we use the identity function of edge labels; i.e., $K_{\text{rel}}(r, r')$ is 1 if the labels of edges r and r' are identical, or else it is 0.

Before proceeding to the experimental procedures and results, we describe how we built the type vectors for words. These type vectors are used for both the WSD and paraphrase ranking experiments.

Construction of context-independent type vectors for words. We first parse the written text part of British National Corpus (BNC) with syntactic-semantic parser Enju[2]. The outputs of Enju are then converted to parse graphs. The edges of the resulting parse graph have one of the following labels: "forward dependency", "backward dependency", "semantic subject (forward semantic subject)", and "semantic predicate (backward semantic subject)." We next collect pairs of (stemmed and lemmatized) words such that (i) each word occurs at least 10 times in BNC, and (ii) the pair is linked by a syntactic/semantic relation at least once in the parse graph collection we converted from BNC, but excluding the pairs for which the *pointwise mutual information* (pmi) (Church and Hanks, 1990) between the words are less than 1. Finally, for

[2] `http://www-tsujii.is.s.u-tokyo.ac.jp/enju/index.html`

every word w occurring in this collection, we make a type vector in which each component corresponds to a pair (w', r) of another word w' and the relation type r between w and w', and holds the pmi score of word w and the pair (w', r),

In preliminary experiments, we also tested type vectors made from the words in a fixed contextual window size (i.e., without using parser outputs), but the results were inferior.

5.1 Experiment 1: word sense disambiguation

For the WSD experiment, we apply the bag-of-walks kernels to the Senseval-3 English lexical sample (ELS) task (Mihalcea et al., 2004).

5.1.1 Task and dataset

The Senseval-3 ELS dataset is a collection of instances of polysemous target words and the contextual paragraphs in which they appear, consisting mostly of several sentences. Each target instance is annotated with one or more gold standard senses selected from a sense inventory (also distributed with the dataset). The dataset comes with predefined training/test splits, and the task goal is to predict a sense for each of the 3944 test instances of 57 polysemous words: 1807 instances for 20 nouns, 1978 for 32 verbs, and 159 for 5 adjectives.

A standard approach for this task is to represent contextual paragraphs as bag-of-words local context vectors. It then predicts the sense of the target word according to rules constructed from training data, usually with a machine learning technique. Recently, to further improve the WSD performance, Giuliano et al. (2009) built a kernel that measures similarity of n-gram collocations containing target words, and combined it with the cosine of local context vectors[3] In this WSD experiment, following Giuliano et al. (2009), we also combine the bag-of-walks kernels with the cosine of local context vectors, and evaluate the performance.

5.1.2 Experimental procedure

WSD with the bag-of-walks kernels consists of four steps: (1) computing the cosine of local context vectors computed from the contextual paragraphs, (2) computing bag-of-walks kernels on parse graphs output by a parser, (3) combining kernels computed in Steps 1 and 2, and (4) sense prediction by support vector machines (SVMs) using the combined kernels. We detail each step below.

Step 1. Compute the cosine of local context vectors. We construct a local context vector, for each target instance according to a standard procedure for WSD (Mihalcea, 2004; Navigli, 2009). Specifically, we treat a contextual paragraph simply as a bag-of-words. After removing stop words[4], we make a local context vector in which components are the weighted frequencies (tf-idf) of words in the "bag." Then we construct a matrix holding cosine between every pairs of local context vectors. Because cosine matrices are positive semi-definite, this matrix can be regarded as a kernel matrix.

Step 2. Compute bag-of-walks kernels. We compute a bag-of-walks kernel between instances of a target word as follows. For each target instance, we pick up a sentence containing the target word from the paragraph given as a context, and then construct a parse graph by

[3] The cosine of local context vectors is called *bag-of-words kernel* in Giuliano et al. (2009).

[4] We use a list of English stop words available from the on-line appendix to (Lewis et al., 2004).

parsing the sentence with the Enju parser. Then we compute the bag-of-walks kernel K using the recursive formula (6) in Section 4.3, for all pairs of target instances. As described in the beginning of this section, K_{word} in Eq. (2) is the cosine similarity between context-independent type vectors, and $K_{\text{rel}}(r, r')$ is the identify kernel of edge labels. Finally, we normalize the obtained bag-of-walks kernel matrix K, so that the (i, j) component of the resulting kernel matrix becomes $K(i, j)/\sqrt{K(i, i)K(j, j)}$.

Step 3. Combine two kernels. In a way similar to Giuliano et al. (2009), we compute a composite kernel by simply adding together the two kernel matrices obtained in Steps 1 and 2.

Step 4. Train SVMs and predict senses of the test data We train multi-class SVMs[5] with the Senseval-3 ELS training data, using the composite kernels obtained in Step 3. After tuning the parameters for each of the polysemous target words by five-fold cross validation on the training set, we train multi-class SVMs with entire training set. The tuned parameters are the walk length bound L and the stopping probability γ of the bag-of-walks kernels, and SVM's soft-margin parameter C. Finally, the trained SVMs are used to predict the sense of the test instances.

5.1.3 Compared methods

We use the most frequent sense prediction as the first baseline. It totally ignores the context, and always predicts the most frequent sense in the training data. This baseline is helpful to evaluate the difficulty of the WSD tasks.

As the second baseline, we use the cosine of local context vectors. Note again that we are interested in how much the performance improves when the bag-of-walks kernels are combined with this simple similarity measure.

We also compare our method with the one proposed by Giuliano et al. (2009), which is also kernel-based. Like bag-of-walks kernels, their kernel takes multi-word co-occurrences with a target word into account, but these co-occurrences are taken in terms of (gap weighted) n-gram collocations; i.e., their kernel counts the overlap of n-gram sequences surrounding the two target words. In contrast, bag-of-walks kernels compute multi-word co-occurrence in the syntactic/semantic structure present in parse graphs. Hence, the comparison of bag-of-walks kernels and Giuliano et al.'s kernel allows us to assess the effectiveness of using syntactic/semantic structure to measure word co-occurrence, rather than using n-grams.[6]

5.1.4 Evaluation

We evaluate predicted senses for test data by F1 score, computed with the scoring script distributed with the Senseval-3 ELS dataset.

5.1.5 Results

Table 1 shows the performance (F1 scores) of the compared methods. The F1 score of Giuliano et al. (2009) is taken from their paper. In total ("ALL"), the proposed method outperforms the

[5] We use the $SVM^{multiclass}$ package: `http://svmlight.joachims.org/svm_multiclass.html`

[6] Giuliano et al. (2009) combined cosine of local context vectors with not only the n-gram collocation kernels, but also "domain kernels." The domain kernels exploit external lexical knowledge sources and singular value decomposition to infer the domain of words. Since our focus is on the effective way of using word co-occurrence in local context, we compare our proposed method only with n-gram collocation kernels (combined with baseline bag-of-words kernels).

Method	ALL	Noun	Verb	Adjective
most frequent sense	55.2	54.2	56.5	49.7
cosine of local context vectors	63.6	64.1	60.3	46.5
proposed method	**71.0**	**72.1**	**71.7**	**50.3**
Giuliano et al. (2009)	69.7	-	-	-

Table 1: F1 scores on the WSD experiment.

baselines and Giuliano et al. (2009)'s method.

These results indicate the effectiveness of using syntactic/semantic structural co-occurrences of multiple words for the WSD tasks.

Table 1 also shows the F1 scores when the test data was broken down by part-of-speech (noun, verb, and adjective). In all parts-of-speech, the proposed method outperformed the baseline. Giuliano et al. (2009) do not report break-down results of their method on individual parts-of-speech, so the corresponding columns are left blank.

The stopping probability γ, and walk length bound L of the bag-of-walks kernels, obtained by cross-validation on the training data, were $\gamma = 0.08$ and $L = 3.0$ on averaged over all the 57 target words. The average value of L for each part-of-speech was $L = 3.6$ for noun, $L = 2.5$ for verb, and $L = 4.2$ for adjective. Thus in all cases, we have $L > 1$, so we conclude that it is worth considering structural co-occurrence of multiple words, not just single words that are directly connected to target words (which corresponds to $L = 1$).

5.2 Experiment 2: ranking paraphrases

In the second experiment, we evaluate bag-of-walks kernels on the task of ranking paraphrases.

5.2.1 Task and dataset

In the paraphrase ranking task, the system is given a target word and the list of its paraphrase candidates. Also given is an instance of the target word with the (sentential) context it appears, and the task goal is to rank the paraphrase candidates by their appropriateness in the light of the given context. No training data is provided, so the system is allowed to use external resources.

Following (Erk and Padó, 2008; Thater et al., 2010; Erk and Padó, 2010; Dinu and Lapata, 2010), we adapt the SemEval-2007 English lexical substitution (ELS) dataset (McCarthy and Navigli, 2007) for a paraphrase ranking task. The dataset consists of 205 target words (59 nouns, 54 verbs, 57 adjectives, and 35 adverbs), each with at most 10 instances of sentential contexts. For each target instance, gold standard paraphrases are annotated and ranked by annotators' votes. Notice that although we use this dataset to organize a paraphrase ranking task, the task setting is different from the original SemEval-2007 ELS task. In SemEval-2007, paraphrase candidates are not provided, and hence the system is required to generate them before ranking. Here in the paraphrase ranking task, we focus solely on ranking candidate paraphrases, which are predefined and given. To make this predefined list of candidate paraphrases from the SemEval-2007 ELS dataset, we pool all the gold standard paraphrases in the dataset for each target word. Now the goal is to rank these candidates for each target instance occurring in a specific context.

Notice that because we use BNC to collect contexts for candidate paraphrases (see Section 5.2.2), paraphrases occurring less than 100 times in the corpus are removed from the candidate lists; a similar filtering was done in (Thater et al., 2010). After this filtering, the average number of candidates for a target word becomes 14.0 in total (13.5 for nouns, 17.6 for verbs, 14.4 for adjectives, and 8.6 for adverbs).

5.2.2 Experimental procedure

Step 1. Collect candidate instances. Given an instance of a target word (i.e., target word in a specific sentential context), we rank its paraphrase candidates according to their contextual similarity to the target instance. To this end, we need the contexts in which candidate words appear, but candidates are provided without context. Hence, for each candidate word, we collect 100 sentential contexts from BNC.

Step 2. Compute bag-of-walks kernels. After collecting sentential contexts for candidate words in Step 1, we convert these sentences into the corresponding parse graphs using the Enju parser and the post-processing described in Section 3.1. The contextual sentences for target instances, which come from SemEval-2007 data, are also converted to parse graphs in the same way. We then apply the bag-of-walks kernels on these graphs to compute the kernel value $K(w, w')$ between a target instance w and each candidate instance w'. The similarity score of w' relative to w is then given by $K(w, w')/\sqrt{K(w, w), K(w', w')}$. In the bag-of-walks kernel computation, we used the same basic kernels K_{word} and K_{rel} as those used for the WSD experiment.

Step 3. Rank candidates. For a given target instance, we compute the score of each candidate word by averaging the similarity scores between the target instance and (a hundred) candidate instances, which we calculated in Step 2. The candidates are then ranked by their scores.

5.2.3 Evaluation

Following previous work (Erk and Padó, 2008; Thater et al., 2010; Erk and Padó, 2010; Dinu and Lapata, 2010), we evaluate the rankings output by each system by *generalized average precision* (GAP) (Kishida, 2005).

5.2.4 Compared methods

The baseline in this task is to rank candidates according to cosine similarity with a target instance using type vectors. Note that since type vectors are context-independent, rankings obtained from this baseline become identical for all instances of a target word regardless of the difference in context in which they appear.

We also compare the performance of bag-of-walks kernels against those reported in two previous studies (Erk and Padó, 2010; Dinu and Lapata, 2010). These studies conducted paraphrase ranking for all target words in the SemEval-2007 dataset, just like this experiment. However, notice that these results are just for reference, because detailed experimental settings, such as the way they built gold candidates for each target word, are probably not the same with ours.

5.2.5 Results

Table 2 shows the performance (GAP score) of the compared methods. For bag-of-walks kernels, we did not tune the parameters since no training data was provided. So the scores displayed

Method	ALL	Noun	Verb	Adjective	Adverb
cosine similarity	44.6	44.0	36.3	45.7	57.0
bag-of-walks kernel ($L = 1$)	46.7	45.2	39.1	47.5	60.1
bag-of-walks kernel ($L = 2$)	47.2	45.5	39.4	**48.7**	60.0
bag-of-walks kernel ($L = 3$)	**47.5**	45.2	40.6	48.6	**60.7**
bag-of-walks kernel ($L = 4$)	47.3	45.6	40.2	**48.7**	59.5
bag-of-walks kernel ($L = 5$)	47.4	45.3	40.8	48.6	59.6
bag-of-walks kernel ($L = 6$)	47.3	45.5	**40.9**	48.1	59.2
bag-of-walks kernel ($L = 7$)	47.0	45.3	40.7	47.6	59.0
bag-of-walks kernel ($L = 8$)	46.8	**45.9**	40.2	47.4	57.7
bag-of-walks kernel ($L = 9$)	46.6	**45.9**	39.9	46.9	57.9
(Erk and Padó, 2010)*	38.6	41.4	38.4	37.5	-
(Dinu and Lapata, 2010)*	42.9	-	-	-	-

Table 2: GAP scores on the paraphrase ranking experiment. Note that the results for the previous studies (*) are just for reference, as the detailed task setting is different from this experiment. Scores for the proposed methods are obtained with $\gamma = 0$.

are those when the walk termination probability γ is set to zero, because it gave the best results. According to Table 2, all the bag-of-walks kernels with maximum walk length L from 1 to 9 outperform the baseline cosine similarity in total ("ALL") as well as when the data is split by part-of-speech (noun, verb, adjective, and adverb). Also, better scores are obtained when $L \geq 3$, indicating that it is effective to take the structural co-occurrences consisting of more than two words into consideration.

Conclusion

We have proposed a new measure of semantic similarity between words in context. It captures structural collocation between the target words and multiple words in the context simultaneously. The proposed measure applies the bag-of-walks kernels, a variation of Desrosiers and Karypis' kernel between graph nodes, to the syntactic/semantic graph output by semantic parsers.

In the experiments on word sense disambiguation and paraphrase ranking, we verified that higher-order relations between words are effective; setting the maximum walk length of $L = 3$ steps achieved the highest performance in these tasks. And in the paraphrase ranking task, if the target words are limited to nouns only, walk length as long as $L = 9$ achieved the highest score.

While we tested our method only on English data, it does not rely on any language-specific features, and hence should work on any languages with sufficiently accurate dependency parsers. Such highly-accurate parsers are now available for many languages, as evidenced by the success of the CoNLL-X shared task (Buchholz and Marsi, 2006).

References

Berger, A. and Lafferty, J. (1999). Information retrieval as statistical translation. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, pages 222–229, Berkeley, California, USA.

Buchholz, S. and Marsi, E. (2006). CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Natural Language Learning (CoNLL-X)*, pages 149–164, New York, NY, USA.

Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Collins, M. and Duffy, N. (2001). Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14 (NIPS '01)*, pages 625–632. MIT Press.

Croce, D., Moschitti, A., and Basili, R. (2011). Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*, pages 1034–1046, Edinburgh, UK.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the Society for Information Science*, 41:391–407.

Desrosiers, C. and Karypis, G. (2009). Within-network classification using local structure similarity. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD '09): Part I*, Lecture Notes in Artificial Intelligence 5781, pages 260–275, Bled, Slovenia. Springer-Verlag.

Dinu, G. and Lapata, M. (2010). Measuring distributional similarity in context. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10)*, pages 1162–1172, Cambridge, Massachusettes, USA.

Erk, K. and Padó, S. (2008). A structured vector space model for word meaning in context. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pages 897–906, Honolulu, Hawaii, USA.

Erk, K. and Padó, S. (2010). Exemplar-based models for word meaning in context. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL '10), Short Papers*, pages 92–97, Uppsala, Sweden.

Gärtner, T. (2003). A survey of kernels for structured data. *SIGKDD Explorations*, 5:49–58.

Giuliano, C., Gliozzo, A. M., and Strapparava, C. (2009). Kernel methods for minimally supervised WSD. *Computational Linguistics*, 35(4):513–528.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA.

Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing*. Pearson Prentice Hall, 2nd edition.

Kashima, H. and Koyanagi, T. (2002). Kernels for semi-structured data. In *Proceedings of the 19th International Conference on Machine Learning (ICML '02)*, pages 291–298, Sydney, Australia.

Kashima, H., Tsuda, K., and Inokuchi, A. (2003). Marginalized kernels between labeled graphs. In *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*, pages 321–328, Washington, DC, USA. AAAI Press.

Kishida, K. (2005). Property of average precision and its generalization: an examination of evaluation indicator for information retrieval experiments. Technical Report NII-2005-014E, National Institute of Informatics, Tokyo, Japan.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. (2004). RCV1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

McCarthy, D. and Navigli, R. (2007). Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53, Prague, Czech Republic.

Mihalcea, R. (2004). Co-training and self-training for word sense disambiguation. In Ng, H. T. and Riloff, E., editors, *Proceedings of the 8th Conference on Computational Natural Language Learning (CoNLL '04)*, pages 33–40, Boston, Massachusetts, USA.

Mihalcea, R., Chklovski, T., and Kilgarriff, A. (2004). The Senseval-3 English lexical sample task. In Mihalcea, R. and Edmonds, P., editors, *Proceedings of the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (Senseval-3)*, pages 25–28, Barcelona, Spain.

Mitchell, J. and Lapata, M. (2008). Vector-based models of semantic composition. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics: Human Language Technologies (ACL '08)*, pages 236–244, Columbus, Ohio, USA.

Moschitti, A., Pighin, D., and Basili, R. (2008). Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys*, 41:10:1–10:69.

Reisinger, J. and Mooney, R. J. (2010). Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT '10)*, pages 109–117, Los Angeles, California, USA.

Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, 24:97–123.

Thater, S., Fürstenau, 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 '10)*, pages 948–957, Uppsala, Sweden.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. (2010). Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242.