Unsupervised Semantic Role Induction with Graph Partitioning

Joel Lang and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB, UK
J.Lang-3@sms.ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

In this paper we present a method for *unsupervised* semantic role induction which we formalize as a graph partitioning problem. Argument instances of a verb are represented as vertices in a graph whose edge weights quantify their role-semantic similarity. Graph partitioning is realized with an algorithm that iteratively assigns vertices to clusters based on the cluster assignments of neighboring vertices. Our method is algorithmically and conceptually simple, especially with respect to how problem-specific knowledge is incorporated into the model. Experimental results on the CoNLL 2008 benchmark dataset demonstrate that our model is competitive with other unsupervised approaches in terms of F1 whilst attaining significantly higher cluster purity.

1 Introduction

Recent years have seen increased interest in the *shallow semantic analysis* of natural language text. The term is most commonly used to describe the automatic identification and labeling of the semantic roles conveyed by sentential constituents (Gildea and Jurafsky, 2002). Semantic roles describe the semantic relations that hold between a predicate and its arguments (e.g., “who” did “what” to “whom”, “when”, “where”, and “how”) abstracting over surface syntactic configurations.

In the example sentences below, *window* occupies different syntactic positions — it is the object of *broke* in sentences (1a,b), and the subject in (1c) — while bearing the same semantic role, i.e., the physical object affected by the breaking event. Analogously, *ball* is the instrument of *break* both when realized as a prepositional phrase in (1a) and as a subject in (1b).

(1) a. [Jim]$_{A0}$ **broke** the [window]$_{A1}$ with a [ball]$_{A2}$.
b. The [ball]$_{A2}$ **broke** the [window]$_{A1}$.
c. The [window]$_{A1}$ **broke** [last night]$_{TMP}$.

The semantic roles in the examples are labeled in the style of PropBank (Palmer et al., 2005), a broad-coverage human-annotated corpus of semantic roles and their syntactic realizations. Under the PropBank annotation framework (which we will assume throughout this paper) each predicate is associated with a set of core roles (named *A*0, *A*1, *A*2, and so on) whose interpretations are specific to that predicate[1] and a set of adjunct roles such as *location* or *time* whose interpretation is common across predicates (e.g., *last night* in sentence (1c)).

The availability of PropBank and related resources (e.g., FrameNet; Ruppenhofer et al. (2006)) has sparked the development of great many semantic role labeling systems most of which conceptualize the task as a supervised learning problem and rely on role-annotated data for model training. Most of these systems implement a two-stage architecture consisting of *argument identification* (determining the arguments of the verbal predicate) and *argument classification* (labeling these arguments with semantic roles). Despite being relatively shallow, se-

[1] More precisely, *A*0 and *A*1 have a common interpretation across predicates as *proto-agent* and *proto-patient* in the sense of Dowty (1991).

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 1320–1331,
Edinburgh, Scotland, UK, July 27–31, 2011.

mantic role analysis has the potential of benefiting a wide spectrum of applications ranging from information extraction (Surdeanu et al., 2003) and question answering (Shen and Lapata, 2007), to machine translation (Wu and Fung, 2009) and summarization (Melli et al., 2005).

Current approaches have high performance — a system will recall around 81% of the arguments correctly and 95% of those will be assigned a correct semantic role (see Màrquez et al. (2008) for details), however only on languages and domains for which large amounts of role-annotated training data are available. For instance, systems trained on PropBank demonstrate a marked decrease in performance (approximately by 10%) when tested on out-of-domain data (Pradhan et al., 2008). Unfortunately, the reliance on role-annotated data which is expensive and time-consuming to produce for every language and domain, presents a major bottleneck to the widespread application of semantic role labeling.

In this paper we argue that unsupervised methods offer a promising yet challenging alternative. If successful, such methods could lead to significant savings in terms of annotation effort and ultimately yield more portable semantic role labelers that require overall less engineering effort. Our approach formalizes semantic role induction as a graph partitioning problem. Given a verbal predicate, it constructs a weighted graph whose vertices correspond to argument instances of the verb and whose edge weights quantify the similarity between these instances. The graph is partitioned into vertex clusters representing semantic roles using a variant of Chinese Whispers, a graph-clustering algorithm proposed by Biemann (2006). The algorithm iteratively assigns cluster labels to graph vertices by greedily choosing the most common label amongst the neighbors of the vertex being updated. Beyond extending Chinese Whispers to the semantic role induction task, we also show how it can be understood as a type of Gibbs sampling when our graph is interpreted as a Markov random field.

Experimental results on the CoNLL 2008 benchmark dataset demonstrate that our method, despite its simplicity, improves upon competitive approaches in terms of F1 and achieves significantly higher cluster purity.

2 Related Work

Although the bulk of previous work on semantic role labeling has primarily focused on supervised methods (Màrquez et al., 2008), a few semi-supervised and unsupervised approaches have been proposed in the literature. The majority of semi-supervised models have been developed within a framework known as *annotation projection*. The idea is to combine labeled and unlabeled data by projecting annotations from a labeled source sentence onto an unlabeled target sentence within the same language (Fürstenau and Lapata, 2009) or across different languages (Padó and Lapata, 2009). Outwith annotation projection, Gordon and Swanson (2007) propose to increase the coverage of PropBank to unseen verbs by finding syntactically similar (labeled) verbs and using their annotations as surrogate training data.

Swier and Stevenson (2004) were the first to introduce an unsupervised semantic role labeling system. Their algorithm induces role labels following a bootstrapping scheme where the set of labeled instances is iteratively expanded using a classifier trained on previously labeled instances. Their method starts with a dataset containing no role annotations at all, but crucially relies on VerbNet (Kipper et al., 2000) for identifying the arguments of predicates and making initial role assignments. VerbNet is a manually constructed lexicon of verb classes each of which is explicitly associated with argument realization and semantic role specifications.

Subsequent work has focused on unsupervised methods for argument identification and classification. Abend et al. (2009) recognize the arguments of predicates by relying solely on part of speech annotations whereas Abend and Rappoport (2010) distinguish between core and adjunct roles, using an unsupervised parser and part-of-speech tagger. Grenager and Manning (2006) address the role induction problem and propose a directed graphical model which relates a verb, its semantic roles, and their possible syntactic realizations. Latent variables represent the semantic roles of arguments and role induction corresponds to inferring the state of these latent variables.

Following up on this work, Lang and Lapata (2010) formulate role induction as the process of detecting alternations and finding a canonical syntactic form for them. Verbal arguments are then assigned roles, according to their position in this canonical form, since each position references a specific role. Their model extends the logistic classifier with hidden variables and is trained in a manner that takes advantage of the close relationship between syntactic functions and semantic roles. More recently, Lang and Lapata (2011) propose a clustering algorithm which first splits the argument instances of a verb into fine-grained clusters based on syntactic cues and then executes a series of merge steps (mainly) based on lexical cues. The split phase creates a large number of small clusters with high purity but low collocation, i.e., while the instances in a particular cluster typically belong to the same role the instances for a particular role are commonly scattered amongst many clusters. The subsequent merge phase conflates clusters with the same role in order to increase collocation.

Like Grenager and Manning (2006) and Lang and Lapata (2010; 2011), this paper describes an unsupervised method for semantic role induction, i.e., one that does not require any role annotated data or additional semantic resources for training. Contrary to these previous approaches, we conceptualize role induction in a novel way, as a graph partitioning problem. Our method is simple, computationally efficient, and does not rely on hidden variables. Moreover, the graph-based representation for verbs and their arguments affords greater modeling flexibility. A wide range of methods exist for finding partitions in graphs (Schaeffer, 2007), besides Chinese Whispers (Biemann, 2006), which could be easily applied to the semantic role induction problem. However, we leave this to future work.

Graph-based methods are popular in natural language processing, especially with unsupervised learning problems (Chen and Ji, 2010). The Chinese Whispers algorithm itself (Biemann, 2006) has been previously applied to several tasks including word sense induction (Klapaftis and M., 2010) and unsupervised part-of-speech tagging (Christodoulopoulos et al., 2010). The same algorithm is also described in Abney (2007, pp. 146-147) under the name "clustering by propagation". The term makes explicit the algorithm's connection to label propagation, a general framework[2] for semi-supervised learning (Zhu et al., 2003) with applications to machine translation (Alexandrescu and Kirchhoff, 2009), information extraction (Talukdar and Pereira, 2010) and structured part-of-speech tagging (Subramanya et al., 2010). The basic idea behind label propagation is to represent labeled and unlabeled instances as vertices in an undirected graph with edges whose weights express similarity (and possibly dissimilarity) between the instances. Label information is then propagated between the vertices in such a way that similar instances tend to be assigned the same label. Analogously, Chinese Whispers works by propagating cluster membership information along the edges of a graph, even though the graph does not contain any human-labeled instance vertices.

3 Problem Setting

We adopt the standard architecture of supervised semantic role labeling systems where argument identification and argument classification are treated separately. Our role labeler is fully unsupervised with respect to both tasks — it does not rely on any role annotated data or semantic resources. However, our system does not learn from raw text. In common with most semantic role labeling research, we assume that the input is syntactically analyzed in the form of dependency trees.

We view argument identification as a syntactic processing step that can be largely undertaken deterministically through structural analysis of the dependency tree. We therefore use a small set of rules to detect arguments with high precision and recall (see Section 4). Argument classification is more challenging and must take into account syntactic *as well as* lexical-semantic information. Both types of information are incorporated into our model through a similarity function that assigns similarity scores to pairs of argument instances. Following previous work (Lang and Lapata, 2010; Grenager and Manning, 2006), our system outputs verb-specific roles by grouping argument instances into clusters and labeling each argument instance with an identifier cor-

[2] For example, Haffari and Sarkar (2007) use label propagation to analyze other semi-supervised algorithms such as the Yarowsky (1995) algorithm.

responding to the cluster it has been assigned to. Such identifiers are similar to PropBank-style core labels (e.g., $A0$, $A1$).

4 Argument Identification

Supervised semantic role labelers often employ a classifier in order to decide for each node in the parse tree whether or not it represents a semantic argument. Nodes classified as arguments are then assigned a semantic role. In the unsupervised setting, we slightly reformulate argument identification as the task of discarding as many non-semantic arguments as possible. This means that the argument identification component does not make a final positive decision for any of the argument candidates; instead, a final decision is only made in the subsequent argument classification stage.

We discard or select argument candidates using the set of rules developed in Lang and Lapata (2011). These are mainly based on the parts of speech and syntactic relations encountered when traversing a dependency tree from the predicate node to the argument node. For each candidate, rules are considered in a prespecified order and the first matching rule is applied. When evaluated on its own, the argument identification component obtained 88.1% precision (percentage of semantic arguments out of those identified) and 87.9% recall (percentage of identified arguments out of all gold arguments).

5 Argument Classification

After identifying likely arguments for each verb, the next step is to infer a label for each argument instance. Since we aim to induce verb-specific roles (see Section 3), we construct an undirected, weighted graph *for each verb*. Vertices correspond to verb argument instances and edge weights quantify the similarities between them. This argument-instance graph is then partitioned into clusters of vertices representing semantic roles and each argument instance is assigned a label that indicates the cluster it belongs to. In what follows we first describe how the graph is constructed and then provide the details of our graph partitioning algorithm.

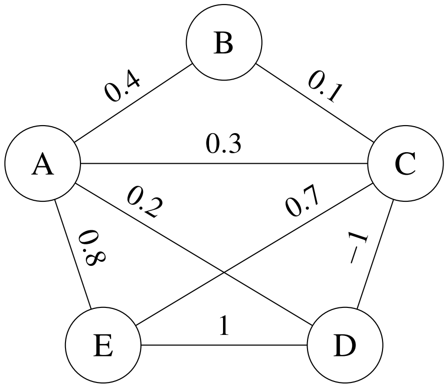

Figure 1: Simplified example of an argument-instance graph. All pairs of vertices with non-zero similarity are connected through edges that are weighted with a similarity score $\phi(v_i, v_j)$. Upon updating the label for a vertex all neighboring vertices propagate their label to the vertex being updated. The score for each label is determined by summing together the weighted votes for that label and the label with the maximal score is chosen.

5.1 Graph Construction

For each verb we construct an undirected, weighted graph $G = (V, E, \phi)$ with vertices V, edges E, and edge weight function ϕ as follows. Each argument instance in the corpus that belongs to the verb is added as a vertex. Then, for each possible pair of vertices (v_i, v_j) we compute a weight $\phi(v_i, v_j) \in \mathbb{R}$ according to the function ϕ. If the weight is non-zero, an undirected edge $e = (v_i, v_j)$ with weight $\phi(v_i, v_j)$ is added to the graph. The function ϕ quantifies the similarity or dissimilarity between instances; positive values indicate that roles are likely to be the same, negative values indicate that roles are likely to differ, and zero values indicate that there is no evidence for either case. Our similarity function is symmetric, i.e., $\phi(v_i, v_j) = \phi(v_j, v_i)$ and permits negative values (see Section 5.4 for a detailed description).

Figure 1 shows an example of a graph for a verb with five argument instances (vertices A–E). Edges are drawn between pairs of vertices with non-zero similarity values. For instance, vertex D is connected to vertex A with weight 0.2, to vertex E with 1, and vertex C with -1. Since edges are drawn between all pairs of vertices with non-zero similarity, the resulting graphs tend to be densely connected, which for large datasets may be prohibitively inefficient. A solution would be to sample a subset from all possible pairs, but we did not make use of any kind of edge pruning in our experiments.

5.2 Graph Partitioning

Graph partitioning is realized with a variant of Chinese Whispers (Biemann, 2006) whose details are given below. In addition, we discuss how our algorithm relates to other graph-based models in order to help provide a better theoretical understanding.

We assume each vertex v_i is assigned a label $l_i \in \{1 \dots L\}$ indicating the cluster it belongs to. Initially, each vertex belongs to its own cluster, i.e., we let the number of clusters $L = |V|$ and set $l_i \leftarrow i$. Given this initial vertex labeling, the algorithm proceeds by iteratively updating the label for each vertex. The update is based on the labels of neighboring vertices and reflects their similarity to the vertex being updated. Intuitively, each neighboring vertex votes for the cluster it is currently assigned to, where the strength of the vote is determined by the similarity (i.e., edge weight) to the vertex being updated. The label l_i of vertex v_i is thus updated according to the following equation:

$$l_i \leftarrow \arg \max_{l \in \{1 \dots L\}} \sum_{v_j \in \mathcal{N}_i(l)} \phi(v_i, v_j) \qquad (2)$$

where $\mathcal{N}_i(l) = \{v_j | (v_i, v_j) \in E \wedge l = l_j\}$ denotes the set of v_i's neighbors with label l. In other words, for each label we compute a score by summing together the weights of edges to neighboring vertices with that label and select the label with the maximal score. Note that negative edges decrease the score for a particular label, thus demoting the label.

Consider again Figure 1. Assume we wish to update vertex A. In addition, assume that B and E are currently assigned the same label (i.e., they belong to the same cluster) whereas C and D are each in different clusters. The score for cluster $\{B, E\}$ is $0.4 + 0.8 = 1.2$, the score for cluster $\{C\}$ is 0.3 and the score for cluster $\{D\}$ is 0.2. We would thus assign A to cluster $\{B, E\}$ as it has the highest score.

The algorithm is run for several iterations. At each iteration it passes over all vertices, and the update order of the vertices is chosen randomly. As the updates proceed, labels can disappear from the graph, whereby the number of clusters decreases. Empirically, we observe that for sufficiently many iterations the algorithm converges to a fixed labeling or oscillates between labelings that differ only in a few vertices. The result of the algorithm is a hard partitioning of the given graph, where the number of clusters is determined automatically.

5.3 Propagation Prioritization

We make one important modification to the basic algorithm described so far based on the intuition that higher scores for a label indicate more reliable propagations. More precisely, when updating vertex v_i to label l we define the confidence of the update as the average similarity to neighbors with label l:

$$conf(l_i \leftarrow l) = \frac{1}{|\mathcal{N}_i(l)|} \sum_{v_j \in \mathcal{N}_i(l)} \phi(v_i, v_j) \qquad (3)$$

We can then prioritize high-confidence updates by setting a threshold θ and allowing only updates with confidence greater or equal to θ. The threshold is initially set to 1 (i.e., the maximal possible confidence) and then lowered by some small constant Δ after each iteration until it reaches a minimum θ_{min}, at which point the algorithm terminates. This improves the resulting clustering, since it promotes reliable updates in earlier phases of the algorithm which in turn has a positive effect on successive updates.

5.4 Argument-Instance Similarity

As described earlier, the edge weights in our graph are similarity scores, with positive values indicating similarity and negative values indicating dissimilarity. Determining the similarity function ϕ without access to labeled training data poses a major difficulty which we resolve by relying on prior linguistic knowledge. Specifically, we measure the similarity of argument instances based on three simple and intuitive criteria: (1) whether the instances are lexically similar; (2) whether the instances occur in the same syntactic position; and (3) whether the instances occur in the same frame (i.e., are arguments in the same clause). The same criteria were used in (Lang and Lapata, 2011) and shown effective in quantifying role-semantic similarity between *clusters* of argument instances. Lexical and syntactic similarity are scored through functions $lex(v_i, v_j)$ and $syn(v_i, v_j)$ with range $[-1, 1]$, whereas the third criterion enters the scoring function directly:

$$\phi(v_i,v_j)=\begin{cases} -\infty \text{ if } v_i \text{ and } v_j \text{ are in same frame} \\ \alpha lex(v_i,v_j)+(1-\alpha)syn(v_i,v_j) \text{ otherwise.} \end{cases} \quad (4)$$

The first case in the function expresses a common linguistic assumption, i.e., that two argument instances v_i and v_j occurring in the same frame cannot have the same semantic role. The function implements this constraint by returning $-\infty$.[3] The syntactic similarity function $s(v_i,v_j)$ indicates whether two argument instances occur in a similar syntactic position. We define syntactic positions through four cues: the relation of the argument head word to its governor, verb voice (active/passive), the linear position of the argument relative to the verb (left/right) and the preposition used for realizing the argument (if any). The score is $\frac{S}{4}$ where S is the number of cues which agree, i.e., have the same value. The syntactic score is set to zero when the governor relation of the arguments is not the same. Lexical similarity $l(v_i,v_j)$ is measured in terms of the cosine of the angle between vectors h_i and h_j representing the argument head words:

$$lex(v_i,v_j)=cos(h_i,h_j)=\frac{h_i \cdot h_j}{||h_i||\,||h_j||} \quad (5)$$

We obtain h_i and h_j from a simple semantic space model (Turney and Pantel, 2010) which requires no supervision (Section 6 describes the details of the model used in our experiments).

Our similarity function weights the contribution of syntax vs. semantics equally, i.e., α is set to 0.5. This reflects the linguistic intuition that lexical and syntactic information are roughly of equal importance.

5.5 Relation to Other Models

This section briefly points out some connections to related models. The averaging procedure used for updating the graph vertices (Equation 2) appears in some form in most label propagation algorithms (see Talukdar (2010) for details). Label propagation algorithms are commonly interpreted as random walks on graphs. In our case such an interpretation is not directly possible due to the presence of negative edge weights. This could be changed by transforming the edge weights onto a non-negative scale, but we find the current setup more expedient for modeling dissimilarity.

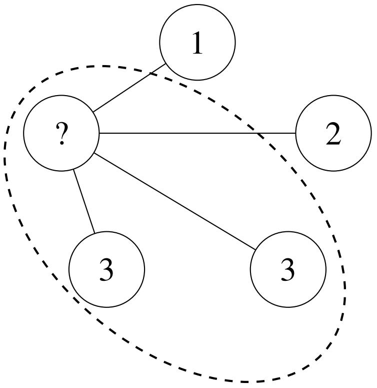

Figure 2: The update rule (Equation 2) can be understood as choosing a minimal edge-cut, thereby greedily maximizing intra-cluster similarity and minimizing inter-cluster similarity. Assuming equal weight for all edges above, label 3 is chosen for the vertex being updated such that the sum of weights of edges crossing the cut is minimal.

Our model could be also transformed into a probabilistic graphical model that specifies a distribution over vertex labels. In the transformed model each vertex corresponds to a random variable over labels and edges are associated with binary potential functions over vertex-pairs. Let $\mathbf{1}(v_i=v_j)$ denote an indicator function which takes value 1 iff. $l_i=l_j$ and value 0, otherwise. Then pairwise potentials can be defined in terms of the original edge weights[4] as $\psi(v_i,v_j)=\exp(\mathbf{1}(v_i=v_j)\phi(v_i,v_j))$. A Gibbs sampler used to sample from the distribution of the resulting pairwise Markov random field (Bishop, 2006; Wainwright and Jordan, 2008) would employ almost the same update procedure as in Equation 2, the difference being that labels would be sampled according to their probabilities, rather than chosen deterministically based on scores.

A third way of understanding the update rule is as a heuristic for maximizing intra-cluster similarity and minimizing inter-cluster similarity. By

[3] Formally, ϕ has range $ran(\phi)=[-1,1]\cup\{-\infty\}$ and for $x\in ran(\phi)$ we define $x+(-\infty)=-\infty$. This means that the overall score computed for a label (Equation 2) is $-\infty$ if one of the summands is $-\infty$.

[4] Including weights with value zero and thus connecting all vertex pairs.

assigning the label with maximal score to v_i, we greedily maximize the sum of intra-cluster edge weights while minimizing the sum of inter-cluster edge weights, i.e., the weight of the edge-cut. This is illustrated in Figure 2. Cut-based methods are a common method in graph clustering (Schaeffer, 2007) and are also used for inference in pairwise Markov random fields like the one described in the previous paragraph (Boykov et al., 2001).

Note that while it would be possible to transform our model into a model with a formal probabilistic interpretation (either as a graph random walk or as a probabilistic graphical model) this would not change the non-empirical nature of the similarity function, which is unavoidable in the unsupervised setting and is also common in the semi-supervised methods discussed in Section 2.

6 Experimental Setup

In this section we describe how we assessed the performance of our model. We discuss the dataset on which our experiments were carried out, explain how our system's output was evaluated and present the methods used for comparison with our approach.

Data We compared the output of our model against the PropBank gold standard annotations contained in the CoNLL 2008 shared task dataset (Surdeanu et al., 2008). The latter was taken from the Wall Street Journal portion of the Penn Treebank and converted into a dependency format (Surdeanu et al., 2008). In addition to gold standard dependency parses, the dataset also contains automatic parses obtained from the MaltParser (Nivre et al., 2007). The dataset provides annotations for verbal and nominal predicate-argument constructions, but we only considered the former, following previous work on semantic role labeling (Màrquez et al., 2008). All the experiments described in this paper use the CoNLL 2008 training dataset.

Evaluation Metrics For each verb, we determine the extent to which argument instances in the clusters share the same gold standard role (purity) and the extent to which a particular gold standard role is assigned to a single cluster (collocation).

More formally, for each group of verb-specific clusters we measure the purity of the clusters as the percentage of instances belonging to the majority gold class in their respective cluster. Let N denote the total number of instances, G_j the set of instances belonging to the j-th gold class and C_i the set of instances belonging to the i-th cluster. Purity can be then written as:

$$PU = \frac{1}{N}\sum_i \max_j |G_j \cap C_i| \tag{6}$$

Collocation is defined as follows. For each gold role, we determine the cluster with the largest number of instances for that role (the role's *primary* cluster) and then compute the percentage of instances that belong to the primary cluster for each gold role:

$$CO = \frac{1}{N}\sum_j \max_i |G_j \cap C_i| \tag{7}$$

Per-verb scores are aggregated into an overall score by averaging over all verbs. We use the micro-average obtained by weighting the scores for individual verbs proportionately to the number of instances for that verb.

Finally, we use the harmonic mean of purity and collocation as a single measure of clustering quality:

$$F_1 = \frac{2 \cdot CO \cdot PU}{CO + PU} \tag{8}$$

Model Parameters Recall that our algorithm prioritizes updates with confidence higher than a threshold θ. Initially, θ is set to 1 and its value decreases at each iteration by a small constant Δ which we set to 0.0025. The algorithm terminates when a minimum confidence θ_{min} is reached. While choosing a value for Δ is straightforward — it simply has to be a small fraction of the maximal possible confidence — specifying θ_{min} on the basis of objective prior knowledge is less so. And although a human judge could determine the optimal termination point based on several criteria such as clustering quality or the number of clusters, we used a development set instead for the sake of reproducibility and comparability. Specifically, we optimized θ_{min} on the CoNLL test set and obtained best results with $\theta_{min} = \frac{1}{3}$. This value was used for all our experiments and was also kept fixed for all verbs. Importantly, the development set was not used for any kind of supervised training.

	Syntactic Function			Latent Logistic			Split-Merge			Graph Partitioning		
	PU	CO	F1	PU	CO	F1	PU	CO	F1	PU	CO	F1
auto/auto	72.9	73.9	73.4	73.2	**76.0**	74.6	81.9	71.2	**76.2**	**82.5**	68.8	75.0
gold/auto	77.7	**80.1**	**78.9**	75.6	79.4	77.4	**84.0**	74.4	**78.9**	**84.0**	73.5	78.4
auto/gold	77.0	71.0	73.9	77.9	**74.4**	76.2	86.5	69.8	**77.3**	**87.4**	65.9	75.2
gold/gold	81.6	**77.5**	79.5	79.5	76.5	78.0	**88.7**	73.0	**80.1**	88.6	70.7	78.6

Table 1: Evaluation of the output of our graph partitioning algorithm compared to our previous models and a baseline that assigns arguments to clusters based on their syntactic function.

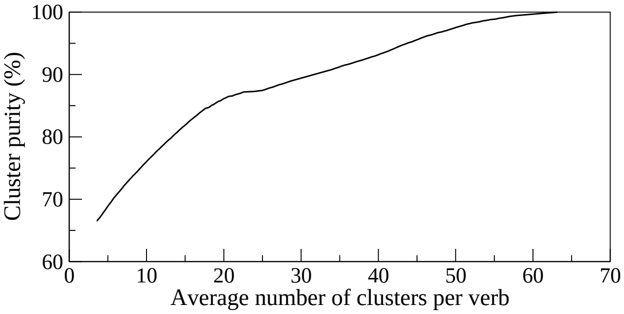

Figure 3: Purity (vertical axis) against average number of clusters per verb (horizontal axis) on the auto/auto dataset.

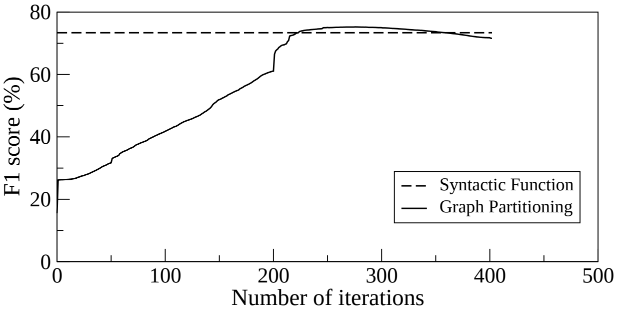

Figure 4: F1 (vertical axis) against number of iterations (horizontal axis) on the auto/auto dataset.

Recall that one of the components in our similarity function is lexical similarity which we measure using a vector-based model (see Section 5.4). We created such a model from the Google N-Grams corpus (Brants and Franz, 2006) using a context window of two words on both sides of the target word and co-occurrence frequencies as vector components (no weighting was applied). The large size of this corpus allows us to use bigram frequencies, rather than frequencies of individual words and to distinguish between left and right bigrams. We used randomized algorithms (Ravichandran et al., 2005) to build the semantic space efficiently.

Comparison Models We compared our graph partitioning algorithm against three competitive approaches. The first one assigns argument instances to clusters according to their syntactic function (e.g., subject, object) as determined by a parser. This baseline has been previously used as a point of comparison by other unsupervised semantic role induction systems (Grenager and Manning, 2006; Lang and Lapata, 2010) and shown difficult to outperform. Our implementation allocates up to $N = 21$ clusters[5] for each verb, one for each of the 20 most frequent syntactic functions and a default cluster for all other functions. We also compared our approach to Lang and Lapata (2010) using the same model settings (with 10 latent variables) and feature set proposed in that paper. Finally, our third comparison model is Lang and Lapata's (2011) split-merge clustering algorithm. Again we used the same parameters and number of clusters (on average 10 per verb). Our graph partitioning method uses identical cues for assessing role-semantic similarity as the method described in Lang and Lapata (2011).

7 Results

Our results are summarized in Table 1. We report cluster purity (PU), collocation (CO) and their harmonic mean (F1) for the baseline (Syntactic Function), our two previous models (the Latent Logistic classifier and Split-Merge) and the graph partitioning algorithm on four datasets. These result from the combination of automatic parses with automatically identified arguments (auto/auto), gold parses with

[5] This is the number of gold standard roles.

Syntactic Function												
PU	91.4	68.6	45.1	59.7	62.4	61.9	63.5	75.9	76.7	69.6	63.1	53.7
CO	91.3	71.9	56.0	68.4	72.7	76.8	65.6	79.7	76.0	63.8	73.4	58.9
F1	91.4	70.2	49.9	63.7	67.1	68.6	64.5	77.7	76.3	66.6	67.9	56.2
Graph Partitioning												
PU	95.6	83.5	72.3	75.4	83.3	84.4	74.8	84.8	89.5	83.0	73.2	66.3
CO	89.1	62.7	42.1	64.2	56.2	66.3	57.2	73.2	64.1	54.3	66.0	57.7
F1	92.2	71.6	53.2	69.4	67.1	74.3	64.8	78.5	74.7	65.7	69.4	61.7
Verb	say	make	go	increase	know	tell	consider	acquire	meet	send	open	break
Freq	15238	4250	2109	1392	983	911	753	704	574	506	482	246

Table 2: Clustering results for individual verbs on the auto/auto dataset with our graph partitioning algorithm and the syntactic function baseline; the scores were taken from a single run.

automatic arguments (gold/auto), automatic parses with gold arguments (auto/gold) and gold parses with gold arguments (gold/gold). Table 1 reports averages across multiple runs. This was necessary in order to ensure that the results of our randomized graph partitioning algorithm are stable.[6] The arguments for the auto/auto and gold/auto datasets were identified using the rules described in Lang and Lapata (2011) (see Section 4). Bold-face is used to highlight the best performing system under each measure (PU, CO, or F1) on each dataset.

Compared to the Syntactic Function baseline, the Graph Partitioning algorithm has higher F1 on the auto/auto and auto/gold datasets but lags behind by 0.5 points on the gold/auto dataset and by 0.9 points on the gold/gold dataset. It attains highest purity on all datasets except for gold/gold, where it is 0.1 points below Split-Merge. When considering F1 in conjunction with purity and collocation, we observe that Graph Partitioning can attain higher purity than the comparison models by trading off collocation. If we were to hand label the clusters output by our system, purity would correspond to the quality of the resulting labeling, while collocation would determine the labeling effort. The relationship is illustrated more explicitly in Figure 3, which plots purity against the average number of clusters per verb on the auto/auto dataset. As the algorithm proceeds the number of clusters is reduced which results in a decrease of purity. The latter decreases more rapidly once the number of 20 clusters per verb is reached. This is accompanied by a decreasing tradeoff ratio between collocation and purity: at this stage decreasing purity by one point increases collocation by roughly one point, whereas in earlier iterations a decrease of purity by one point goes together with several points increase in collocation. This is most likely due to the fact that the number of gold standard classes is around 20.

Figure 4 shows the complete learning curve of our graph partitioning method on the auto/auto dataset (F1 is plotted against the number of iterations). The algorithm naturally terminates at iteration 266 (when $\theta_{min} = 1/3$), but we have also plotted iterations beyond that point. Since lower values of θ permit unreliable propagations, F1 eventually falls below the baseline (see Section 5.2). The importance of our propagation prioritization mechanism is further underlined by the fact that when it is not employed (i.e., when using the vanilla Chinese Whispers algorithm without any modifications), it performs substantially worse than the comparison models. On the auto/auto dataset, F1 converges to 59.1 (purity is 55.5 and collocation 63.2) within 10 iterations.

Finally, Table 2 shows how performance varies across verbs. We report results for the Syntactic Function baseline and Graph Partitioning on the auto/auto dataset for 12 verbs. These were selected so as to exhibit varied occurrence frequencies and alternation patterns. As can be seen, the macro-

[6] For example, on the auto/auto dataset and over 10 runs, the standard deviation in F1 was 0.11 points in collocation 0.16 points and in purity 0.08 points. The worst F1 was 0.20 points below the average, the worst collocation was 0.32 points below the average and the worst purity was 0.17 points below the average.

scopic result — increase in F1 and purity — also holds across verbs.

8 Conclusions

In this paper we described an unsupervised method for semantic role induction, in which argument-instance graphs are partitioned into clusters representing semantic roles. The approach is conceptually and algorithmically simple and novel in its formalization of role induction as a graph partitioning problem. We believe this constitutes an interesting alternative for two reasons. Firstly, eliciting and encoding problem-specific knowledge in the form of instance-wise similarity judgments can be easier than encoding it into model structure e.g., by making statistical independence assumptions or assumptions about latent structure. Secondly, the approach is general and amenable to other graph partitioning algorithms and relates to well-known graph-based semi-supervised learning methods.

The similarity function in this paper is by necessity rudimentary, since it cannot be estimated from data. Nevertheless, the resulting system attains competitive F1 and notably higher purity than the comparison models. Arguably, performance could be improved by developing a better similarity function. Therefore, in the future we intend to investigate how our system performs in a weakly supervised setting, where the similarity function is estimated from a small amount of labeled instances, since this would allow us to incorporate richer syntactic features and result in more precise similarity scores.

Acknowledgments We are grateful to Charles Sutton for his valuable feedback on this work. The authors acknowledge the support of EPSRC (grant GR/T04540/01).

References

O. Abend and A. Rappoport. 2010. Fully unsupervised core-adjunct argument classification. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 226–236, Uppsala, Sweden.

O. Abend, R. Reichart, and A. Rappoport. 2009. Unsupervised Argument Identification for Semantic Role Labeling. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 28–36, Singapore.

S. Abney. 2007. *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC.

A. Alexandrescu and K. Kirchhoff. 2009. Graph-based learning for statistical machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 119–127, Boulder, Colorado.

C. Biemann. 2006. Chinese Whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80, New York City.

C. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Y. Boykov, O. Veksler, and R. Zabih. 2001. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239.

T. Brants and A. Franz. 2006. Web 1T 5-gram Version 1. Linguistic Data Consortium, Philadelphia.

Z. Chen and H. Ji. 2010. Graph-based clustering for computational linguistics: A survey. In *Proceedings of TextGraphs-5 - 2010 Workshop on Graph-based Methods for Natural Language Processing*, pages 1–9, Uppsala, Sweden.

C. Christodoulopoulos, S. Goldwater, and M. Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 575–584, Cambridge, MA.

D. Dowty. 1991. Thematic Proto Roles and Argument Selection. *Language*, 67(3):547–619.

H. Fürstenau and M. Lapata. 2009. Graph Aligment for Semi-Supervised Semantic Role Labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 11–20, Singapore.

D. Gildea and D. Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

A. Gordon and R. Swanson. 2007. Generalizing Semantic Role Annotations Across Syntactically Similar Verbs. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 192–199, Prague, Czech Republic.

T. Grenager and C. Manning. 2006. Unsupervised Discovery of a Statistical Verb Lexicon. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 1–8, Sydney, Australia.

G. Haffari and A. Sarkar. 2007. Analysis of Semi-Supervised Learning with the Yarowsky Algorithm. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, Vancouver, BC.

K. Kipper, H. T. Dang, and M. Palmer. 2000. Class-Based Construction of a Verb Lexicon. In *Proceedings of the 17th AAAI Conference on Artificial Intelligence*, pages 691–696. AAAI Press / The MIT Press.

I. Klapaftis and Suresh M. 2010. Word sense induction & disambiguation using hierarchical random graphs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 745–755, Cambridge, MA.

J. Lang and M. Lapata. 2010. Unsupervised Induction of Semantic Roles. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947, Los Angeles, California.

J. Lang and M. Lapata. 2011. Unsupervised Semantic Role Induction via Split-Merge Clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, Portland, Oregon. To appear in.

L. Màrquez, X. Carreras, K. Litkowski, and S. Stevenson. 2008. Semantic Role Labeling: an Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159.

G. Melli, Y. Wang, Y. Liu, M. M. Kashani, Z. Shi, B. Gu, A. Sarkar, and F. Popowich. 2005. Description of SQUASH, the SFU Question Answering Summary Handler for the DUC-2005 Summarization Task. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing Document Understanding Workshop*, Vancouver, Canada.

J. Nivre, J. Hall, J. Nilsson, G. Eryigit A. Chanev, S. Kübler, S. Marinov, and E. Marsi. 2007. Malt-Parser: A Language-independent System for Data-driven Dependency Parsing. *Natural Language Engineering*, 13(2):95–135.

S. Padó and M. Lapata. 2009. Cross-lingual Annotation Projection of Semantic Roles. *Journal of Artificial Intelligence Research*, 36:307–340.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

S. Pradhan, W. Ward, and J. Martin. 2008. Towards Robust Semantic Role Labeling. *Computational Linguistics*, 34(2):289–310.

D. Ravichandran, P. Pantel, and E. Hovy. 2005. Randomized Algorithms and NLP: Using Locality Sensitive Hash Function for High Speed Noun Clustering. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, page 622629, Ann Arbor, Michigan.

J. Ruppenhofer, M. Ellsworth, M. Petruck, C. Johnson, and J. Scheffczyk. 2006. FrameNet II: Extended Theory and Practice, version 1.3. Technical report, International Computer Science Institute, Berkeley, CA, USA.

S. Schaeffer. 2007. Graph clustering. *Computer Science Review*, 1(1):27–64.

D. Shen and M. Lapata. 2007. Using Semantic Roles to Improve Question Answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning*, pages 12–21, Prague, Czech Republic.

A. Subramanya, S. Petrov, and F. Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176, Cambridge, MA.

M. Surdeanu, S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using Predicate-Argument Structures for Information Extraction. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 8–15, Sapporo, Japan.

M. Surdeanu, R. Johansson, A. Meyers, and L. Màrquez. 2008. The CoNLL-2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *Proceedings of the 12th CoNLL*, pages 159–177, Manchester, England.

R. Swier and S. Stevenson. 2004. Unsupervised Semantic Role Labelling. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing*, pages 95–102, Barcelona, Spain.

P. Talukdar and F. Pereira. 2010. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1473–1481, Uppsala, Sweden.

P. Talukdar. 2010. *Graph-Based Weakly Supervised Methods for Information Extraction & Integration*. Ph.D. thesis, CIS Department, University of Pennsylvania.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

M. Wainwright and M. Jordan. 2008. Graphical Models, Exponential Families, and Variational Inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.

D. Wu and P. Fung. 2009. Semantic Roles for SMT: A Hybrid Two-Pass Model. In *Proceedings of North*

American Annual Meeting of the Association for Computational Linguistics HLT 2009: Short Papers, pages 13–16, Boulder, Colorado.

D. Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, MA.

X. Zhu, Z. Ghahramani, and J. Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the 20th International Conference on Machine Learning*, Washington, DC.