Similarity-Driven Semantic Role Induction via Graph Partitioning

Joel Lang*
University of Geneva

Mirella Lapata**
University of Edinburgh

As in many natural language processing tasks, data-driven models based on supervised learning have become the method of choice for semantic role labeling. These models are guaranteed to perform well when given sufficient amount of labeled training data. Producing this data is costly and time-consuming, however, thus raising the question of whether unsupervised methods offer a viable alternative. The working hypothesis of this article is that semantic roles can be induced without human supervision from a corpus of syntactically parsed sentences based on three linguistic principles: (1) arguments in the same syntactic position (within a specific linking) bear the same semantic role, (2) arguments within a clause bear a unique role, and (3) clusters representing the same semantic role should be more or less lexically and distributionally equivalent. We present a method that implements these principles and formalizes the task as a graph partitioning problem, whereby argument instances of a verb are represented as vertices in a graph whose edges express similarities between these instances. The graph consists of multiple edge layers, each one capturing a different aspect of argument-instance similarity, and we develop extensions of standard clustering algorithms for partitioning such multi-layer graphs. Experiments for English and German demonstrate that our approach is able to induce semantic role clusters that are consistently better than a strong baseline and are competitive with the state of the art.

1. Introduction

Recent years have seen increased interest in the **shallow semantic analysis** of natural language text. The term is often used to describe the automatic identification and labeling of the semantic roles conveyed by sentential constituents (Gildea and Jurafsky 2002). Semantic roles describe the 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. This type of semantic information

* Department of Computer Science, University of Geneva, 7 route de Drize, 1227 Carouge, Switzerland, E-mail: Joel.Lang@unige.ch.
** Institute for Language, Cognition and Computation, School of Informatics, University of Edinburgh, 10 Crichton Street, EH8 9AB, E-mail: mlap@inf.ed.ac.uk.

Submission received: 26 December 2012; revised version received: 19 September 2013; accepted for publication: 20 November 2013.

doi:10.1162/COLI_a_00195

is shallow but relatively straightforward to infer automatically and useful for the development of broad-coverage, domain-independent language understanding systems. Indeed, the analysis produced by existing semantic role labelers has been shown to benefit 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).

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). In all instances, it bears the same semantic role, that is, the *patient* or 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. $[\text{Jim}]_{\text{A0}}$ **broke** the $[\text{window}]_{\text{A1}}$ with a $[\text{ball}]_{\text{A2}}$.

b. The $[\text{ball}]_{\text{A2}}$ **broke** the $[\text{window}]_{\text{A1}}$.

c. The $[\text{window}]_{\text{A1}}$ **broke** $[\text{last night}]_{\text{TMP}}$.

Also notice that all three instances of *break* in Example (1) have apparently similar surface syntax with a subject and a noun directly following the predicate. However, in sentence (1a) the subject of *break* expresses the *agent* role, in (1b) it expresses the *instrument* role, and in (1c) the *patient* role.

The examples illustrate the fact that predicates can license several alternate mappings or **linkings** between their semantic roles and their syntactic realization. Pairs of linkings allowed by a single predicate are often called **diathesis alternations** (Levin 1993). Sentence pair (1a,b) is an example of the instrument subject alternation, and pair (1b,c) illustrates the causative alternation. Resolving the mapping between the syntactic dependents of a predicate (e.g., *subject*, *object*) and the semantic roles that they each express is one of the major challenges faced by semantic role labelers.

The semantic roles in the examples are labeled in the style of PropBank (Palmer, Gildea, and Kingsbury 2005), a broad-coverage human-annotated corpus of semantic roles and their syntactic realizations. Under the PropBank annotation framework each predicate is associated with a set of core roles (named A0, A1, A2, 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 a variety 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). Current approaches deliver reasonably good 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), although only on languages and domains for which large amounts of role-annotated training data are available.

Unfortunately, the reliance on labeled data, which is both difficult and expensive to produce, presents a major obstacle to the widespread application of semantic role labeling across different languages and text genres. Although corpora with semantic

1 More precisely, A0 and A1 have a common interpretation across predicates as *proto-agent* and *proto-patient* in the sense of Dowty (1991).

role annotations exist nowadays in other languages (e.g., German, Spanish, Catalan, Chinese, Korean), they tend to be smaller than their English equivalents and of limited value for modeling purposes. Even within English, a language for which two major annotated corpora are available, systems trained on PropBank demonstrate a marked decrease in performance (approximately by 10%) when tested on out-of-domain data (Pradhan, Ward, and Martin 2008). The data requirements for supervised systems and the current paucity of such data has given impetus to the development of unsupervised methods that learn from unlabeled data. If successful, unsupervised approaches could lead to significant resource savings and the development of semantic role labelers that require less engineering effort. Besides being interesting on their own right, from a theoretical and linguistic perspective, unsupervised methods can provide valuable features for downstream (supervised) processing and serve as a preprocessing step for applications that require broad coverage understanding. In this article we study the potential of unsupervised methods for semantic role labeling. As in the supervised case, we decompose the problem into an argument identification step and an argument classification step. Our work primarily focuses on argument classification, which we term **role induction**, because there is no predefined set of semantic roles in the unsupervised case, and these must be induced from data. The goal is to assign argument instances to clusters such that each cluster contains arguments corresponding to a specific semantic role and each role corresponds to exactly one cluster.

Unsupervised learning is known to be challenging for many natural language processing problems and role induction is no exception. Firstly, it is difficult to define a learning objective function whose optimization will yield an accurate model. This contrasts with the supervised setting, where the objective function can directly reflect training error (i.e., some estimate of the mismatch between model output and the gold standard) and the model can be tuned to replicate human output for a given input under mathematical guarantees regarding the accuracy of the trained model. Secondly, it is also more difficult to incorporate rich feature sets into an unsupervised model (Berg-Kirkpatrick et al. 2010). Unless we explicitly know exactly how features interact, more features may not necessarily lead to a more accurate model and may even decrease performance. In the supervised setting, feature interactions relevant for a particular learning task can be determined to a large extent automatically and thus a large number of them can be included even if their significance is not clear a priori.

The lack of an extensional definition (in the form of training examples) of the target concept makes a strong case for the development of unsupervised methods that use problem specific prior knowledge. The idea is to derive a strong inductive bias (Gordon and Desjardins 1995) based on this prior knowledge that will guide the learning towards the correct target concept. For semantic role induction, we propose to build on the following linguistic principles:

1. Semantic roles are unique within a particular frame.
2. Arguments occurring in a specific syntactic position *within a specific linking* all bear the same semantic role.
3. The (asymptotic) distribution over argument heads is the same for two clusters that represent the same semantic role.

We hypothesize that these three principles are, at least in theory, sufficient for inducing high-quality semantic role clusters. A challenge, of course, lies in adequately operationalizing them so that they guide the unsupervised learner towards meaningful

solutions. The approach taken in this article translates these principles into estimates of similarity (or dissimilarity) between argument instances and/or clusters of argument instances. Principle (1) states that argument instances occurring in the same frame (i.e., clause) cannot bear the same semantic role, and are thus dissimilar. From Principle (2) it follows that arguments occurring in the same syntactic position within the same linking can be considered similar (leaving aside for the moment the difficulty of representing linkings through syntactic cues observable in a corpus). Principle (3) states that two clusters of instances containing similar distributions over head words should be considered similar.

Based on these similarity estimates we construct a graph whose vertices represent argument instances and whose edges express similarities between these instances. The graphs consist of multiple edge layers, each capturing one particular type of argument-instance similarity. For example, one layer will be used to represent whether argument instances occur in the same frame, and another layer will represent whether two arguments have a similar head word, and so on. Given this graph representation of the data, we formalize role induction as the problem of partitioning the graph into clusters of similar vertices. We present two algorithms for partitioning multi-layer graphs, which are adaptations of standard graph partitioning algorithms to the multi-layer setting. The algorithms differ in the way they exploit the similarity information encoded in the graph. The first one is based on *agglomeration*, where two clusters containing similar instances are grouped into a larger cluster. The second one is based on *propagation*, where role-label information is transferred from one cluster to another based on their similarity.

To understand how the aforementioned principles might allow us to handle the ambiguity stemming from alternate linkings, consider again Example (1). The most important thing to note is that, whereas the *subject* position is ambiguous with respect to the semantic roles it can express (it can be A0, A1, or A2), we can resolve the ambiguity by exploiting overt syntactic cues of the underlying linking. For example, the predicate *break* is transitive in sentences (1a) and (1b), and intransitive in sentence (1c). Thus, by taking into account the argument's syntactic position *and* the predicate's transitivity, we can guess that the semantic role expressed by the subject in sentence (1c) is different from the roles expressed by the subjects in sentences (1a,b). Now consider the more difficult case of distinguishing between the subjects in sentences (1a) and (1b). One linking cue that could help here is the prepositional phrase in sentence (1a), which results in a syntactic frame different from sentence (1b). Were the prepositional phrase omitted, we would attempt to disambiguate the linkings by resorting to lexical-semantic cues (e.g., by taking into account whether the subject is animate). In sum, if we encode sufficiently many linking cues, then the resulting fine-grained syntactic information will discriminate ambiguous semantic roles. In cases where syntactic cues are not discerning enough, we can exploit lexical information and group arguments together based on their lexical content.

The remainder of this article is structured as follows. Section 2 provides an overview of unsupervised methods for semantic role labeling. Sections 3 and 4 present the details of our method, that is, how the graphs are constructed and partitioned. Role induction experiments in English and German are described in sections 5 and 6, respectively. Discussion of future work concludes in section 7.

2. Related Work

The bulk of previous work on semantic role labeling has focused on supervised methods (Màrquez et al. 2008), although a few semi-supervised and unsupervised approaches

have been proposed. 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). Beyond 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 data set containing no role annotations at all, but crucially relies on VerbNet (Kipper, Dang, and Palmer 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.

In this article we will not assume the availability of any role-semantic resources, although we do assume that sentences are syntactically analyzed. There have been two main approaches to role induction from parsed data. Under the first approach, semantic roles are modeled as latent variables in a (directed) graphical model that relates a verb, its semantic roles, and their possible syntactic realizations (Grenager and Manning 2006). Role induction here corresponds to inferring the state of the latent variables representing the semantic roles of arguments. Following up on this work, Lang and Lapata (2010) reformulate 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, because 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, Garg and Henderson (2012) extend the latent-variable approach by modeling the sequential order of roles.

The second approach is similarity-driven and based on clustering. Lang and Lapata (2011a) propose an algorithm that first splits the set of all argument instances of a verb according to their syntactic position within a particular linking and then iteratively merges clusters. A different clusstering algorithm is adopted in Lang and Lapata (2011b). Specifically, they induce semantic roles via graph partitioning: Each vertex in the graph corresponds to an argument instance and edges represent a heuristically defined measure of their lexical and syntactic similarity. The similarity-driven approach has been recently adopted by Titov and Klementiev (2012a), who propose a Bayesian clustering algorithm based on the Chinese Restaurant Process. In addition, they present a method that shares linking preferences across verbs using a distance-dependent Chinese Restaurant Process prior which encourages similar verbs to have similar linking preferences. Titov and Klementiev (2012b) further introduce the use of multilingual data for improving role induction.

There has also been work on unsupervised methods for argument identification. Abend, Reichart, and Rappoport (2009) devise a method for recognizing the arguments of predicates that relies solely on part of speech annotations, whereas Abend and Rappoport (2010a) distinguish between core and adjunct roles, using an unsupervised parser and part-of-speech tagger. More generally, shallow semantic representations

induced from syntactic information are commonly used in lexicon acquisition and information extraction tasks. For example, Lin and Pantel (2001) cluster syntactic relations between pairs of words as expressed by parse tree paths into semantic relations by exploiting lexical distributional similarity. Although not compatible with PropBank or semantic roles as such, Poon and Domingos (2009) and Titov and Klementiev (2011) also induce semantic information from dependency parses and apply it to a question answering task for the biomedical domain. Another example is the work by Gamallo, Agustini, and Lopes (2005), who cluster similar syntactic positions in order to develop models of selectional preferences to be used for word sense induction and the resolution of attachment ambiguities.

The work described here unifies the two clustering methods presented in Lang and Lapata (2011a and 2011b) by reformulating them as graph partitioning algorithms. It also extends them by utilizing *multi-layer* graphs which separate the similarities between instances on different features (e.g., part-of-speech, argument head) into different layers. This has the advantage that similarity scores on individual features do not have to be eagerly combined into a similarity score *between instances*. Instead, one can first aggregate the similarity scores on each feature layer between two clusters and then combine them into a similarity score between *clusters*. This is more robust, as the feature-wise similarity scores between clusters can be computed in a principled way and the heuristic combination step is deferred to the end (see Section 4 for details). Besides providing a general modeling framework for semantic role induction, we discuss in detail the linguistic principles guiding our modeling choices and assess their applicability across languages. Specifically, we show that the framework presented here (and the aforementioned principles) can be readily applied to English and German with identical parametrizations for both languages and without fundamentally changing the underlying model features, despite major syntactic differences between the two languages.

3. Graph Construction

We begin by explaining how we construct a graph that represents verbs and their arguments. Next, we describe how edge weights are computed—these translate to similarity scores between argument instances—and then move on to provide the details of our graph-partitioning algorithms.

As mentioned earlier, we formalize semantic role induction as a clustering problem. Clustering algorithms (see Jain, Murty, and Flynn [1999] for an overview) commonly take a matrix of pairwise similarity scores between instances as input and produce a set of output clusters, often satisfying some explicitly defined optimality criterion. The success or failure of the clustering approach is closely tied to the adequacy of the employed similarity function for the task at hand. The graph partitioning view of clustering (see Schaeffer [2007] for a detailed treatment) arises when instances are represented as the vertices of a graph and the similarity matrix is interpreted as the weight matrix of the graph. For semantic role induction, a straightforward application of clustering would be to construct a graph for each verbal predicate such that vertices correspond to argument instances of the verb and edge weights quantify the similarity between these instances.

Lang and Lapata (2011b) hand-craft an instance similarity function by taking into account different features such as the argument head or its syntactic position. Defining an appropriate instance-wise similarity function is nevertheless problematic as

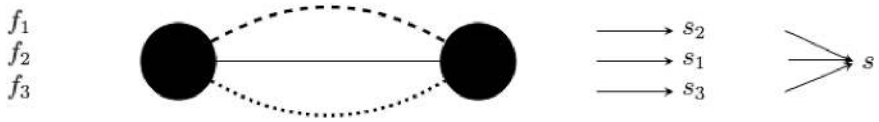

Figure 1
A multi-layer graph consists of multiple edge layers, one for each similarity feature. Multi-layer graph partitioning algorithms exploit this representation by computing separate similarity scores between clusters for each feature layer and then combining them into a single overall similarity score. This is advantageous over single-layer graph partitioning because it avoids eagerly combining the similarity scores for individual features into a heuristic instance-wise similarity score.

weights have to be chosen heuristically. Instead, we will represent similarities with respect to different *features* on separate edge layers in the graph. For example, one layer will represent the similarity between the head words of arguments and another one will represent the similarity between pars of speech. So, given M features, the graph will consist of M layers, one for each feature. Edge weights on a particular layer quantify the similarity between the instances with respect to that feature. This is illustrated in Figure 1 for two argument instances and three features. Formally, a multi-layer graph is defined as a pair $(V, \{E_1, \ldots, E_M\})$ consisting of vertices V and a set of edge layers E_f for $f = 1 \ldots M$. The set of vertices $V = \{v_1, \ldots, v_N\}$ consists of all N argument instances for a particular verb. The edge layer E_f for feature f is constructed by connecting all vertex-pairs with non-zero similarity with respect to f:

$$E_f = \{(v_i, v_j) \in V \times V | \phi_f(v_i, v_j) \neq 0\}. \tag{2}$$

where $\phi_f(v_i, v_j)$ is a similarity function for feature f, whose form will be discussed in the next section. Each edge $(v_i, v_j) \in E_f$ in layer f is weighted by $\phi_f(v_i, v_j)$.

3.1 Feature Similarity Functions

Similarities for a specific feature f are measured with a function $\phi_f(v_i, v_j)$ which assigns a $[-1, 1]$ value to any pair of instances (v_i, v_j). We assume similarities are measured on an interval scale—that is, while sums, differences, and averages of the values of some similarity function ϕ_f express meaningful quantities, products and ratios do not. Moreover, the values of two distinct similarity functions cannot necessarily be meaningfully compared without rescaling. Positive similarity values indicate that the semantic 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. The magnitude of ϕ_f expresses the degree of confidence in the similarity judgment, with extreme values (i.e., -1 and 1) indicating maximal confidence.

In our model, we simply use indicator functions which output either 1 or -1 iff feature values are equal and 0 otherwise. Specifically, we define four feature similarity functions that we derive from the principles discussed in Section 1. Our similarity functions are based on the following features: the argument head words and their parts

of speech,[2] the frame constraint, and the syntactic position within a particular linking. We measure lexical and part-of-speech similarity as follows:

$$\phi_{lex}(v_i, v_j) = \begin{cases} 1 & \text{if } v_i^{lex} = v_j^{lex} \\ 0 & \text{otherwise} \end{cases} \qquad \phi_{pos}(v_i, v_j) = \begin{cases} 1 & \text{if } v_i^{pos} = v_j^{pos} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The constraint that two argument instances v_i and v_j occurring in the same frame cannot have the same semantic role is captured by the following similarity function:

$$\phi_{frame}(v_i, v_j) = \begin{cases} -1 & \text{if } v_i^{frame} = v_j^{frame} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Finally, we also measure syntactic similarity through an indicator function $\phi_{syn}(v_i, v_j)$, which assumes value 1 if two instances occur in the same syntactic position within the same linking:

$$\phi_{syn}(v_i, v_j) = \begin{cases} 1 & \text{if } v_i^{syn} = v_j^{syn} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The syntactic position of an argument is directly given by the parse tree and can be encoded, for example, by the full path from predicate to argument head, or for practical purposes, in order to reduce sparsity, simply through the relation governing the argument head and its linear position relative to the predicate (left or right). In contrast, linkings are not directly observed, but we can resort to overt syntactic cues as a proxy. Examples include the verb's voice (active/passive), whether it is transitive, the part-of-speech of the subject, and so on. We argue that in principle, if sufficiently many cues are taken into account, they will capture one particular linking, although there may be several encodings for the same linking. Note that syntactic similarity is not used to construct another graph layer; rather, it will be used for deriving initial clusters of instances, as we explain in Section 4.1.

4. Graph Partitioning

The graph partitioning problem consists of finding a set of clusters $\{c_1, \ldots, c_S\}$ that form a partition of the vertex-set, namely, $\cup_i c_i = V$ and $c_i \cap c_j = \emptyset$ for all $i \neq j$, such that (ideally) each cluster contains argument instances of only one particular semantic role, and the instances for a particular role are all assigned to one and the same cluster. In the following sections we provide two algorithms for multi-layer graph partitioning, based on standard clustering algorithms for single-layer graphs. Both algorithms operate on the *same graph* but differ in terms of the underlying clustering mechanism they use. The first algorithm is an adaptation of agglomerative clustering (Jain, Murty, and Flynn 1999) to the multi-layer setting: Starting from an initial clustering, the algorithm

2 We include parts of speech as a simple means of alleviating the sparsity of head words.

Algorithm 1 Cluster merging procedure. Operation $merge(c_i, c_j)$ merges cluster c_i into cluster c_j and removes c_i from the list C.

```
1  while not done do
2  |  C ← a list of all clusters sorted by number of instances in descending order
3  |  i ← 1
4  |  while i < length(C) do
5  |  |  j ← arg max_{0≤j'<i} s(c_i, c_j')
6  |  |  if s(c_i, c_j) > 0 then
7  |  |  |  merge(c_i, c_j)
8  |  |  end
9  |  |  else
10 |  |  |  i ← i + 1
11 |  |  end
12 |  end
13 |  update-thresholds
14 end
```

iteratively merges vertex clusters in order to arrive at increasingly accurate representations of semantic roles. Rather than greedily merging clusters, our second algorithm is based on propagating cluster membership information among the set of initial clusters (Abney 2007).

4.1 Agglomerative Graph Partitioning

The agglomerative algorithm induces clusters in a bottom–up manner starting from an initial cluster assignment that we will subsequently discuss in detail. Our initialization results in a clustering that has high purity but low collocation, that is, argument instances in each cluster tend to belong to the same role but argument instances of a particular role are scattered among many clusters.[3] The algorithm then improves collocation by iteratively merging pairs of clusters. The agglomeration procedure is described in Algorithm 1. As can be seen, pairs of clusters are merged iteratively until a termination criterion is met. The decision of which cluster pair to merge at each step is made by scoring a set of candidate cluster pairs and choosing the highest one (line 5). The scoring function $s(c_i, c_{j'})$ quantifies how likely two clusters are to contain arguments of the same role. A key question is how to define this scoring function on the basis of the underlying graph representation, that is, with reference to the instance similarities expressed by the edges. In order to collect evidence for or against a merge, we take into account the connectivity of a cluster pair at each feature layer of the graph. This crucially involves aggregating over all edges that connect the two clusters, and allows us to infer a cluster-level similarity score from the individual instance-level similarities encoded in the edges. The evidence collected at each layer is then combined together in order to arrive at an overall decision (see Figure 1 for an illustration).

3 We define the terms *purity* and *collocation* more formally in Section 5.4.

Although it would be possible to enumerate and score all possible cluster pairs at each step, we apply a more efficient and effective procedure in which the set of candidates consists of pairs formed by combining a fixed cluster c_i with all clusters c'_j larger than c_i. This requires comparing only $O(|C|)$ rather than $O(|C|^2)$ scores and, more importantly, it favors merges between large clusters whose score can be computed more reliably. As mentioned earlier, our scoring function implements an averaging procedure over the instances contained in the clusters, and thus yields less noisy scores when clusters are large (i.e., contain many instances). This prioritization promotes reliable merges over less reliable ones in the earlier phases of the algorithm with a positive effect on merges in the later phases. Moreover, by keeping c_i fixed, we only require that scores $s(c_i, x)$ and $s(c_i, z)$ are comparable (i.e., where one cluster is argument in both scores), rather than comparisons between arbitrary cluster pairs (e.g., $s(w, x)$ and $s(y, z)$). In the following, we will provide details on the initialization of the algorithm and the computation of the similarity scoring function.

A standard agglomerative clustering algorithm forms clusters bottom–up by initially placing each item of interest in its own cluster. In our case, initializing the algorithm with as many clusters as argument instances would result in a clustering with maximal purity and minimal collocation. There are two reasons that justify a more sophisticated initialization procedure for our problem. Firstly, the scoring function we use is more reliable for larger clusters than for smaller clusters (see the subsequent discussion). In fact, the standard initialization that creates clusters with a single instance would not yield useful results as our scoring function crucially relies on initial clusters containing several instances on average. Secondly, the similarity scores for different features are not directly comparable. Recall from Section 3.1 that we introduced different types of similarities based on the arguments' head words (ϕ_{lex}), parts-of-speech (ϕ_{pos}), syntactic positions (ϕ_{syn}), and frame constraints (ϕ_{frame}). As discussed earlier, engineering a scoring function that integrates these into a single score without resorting to heuristic judgments on how to weight them poses a major challenge. In particular, it is difficult to weight the contribution of the two forms of positive evidence given by lexical and syntactic similarity. This motivates the idea of using syntactic similarity for initialization, and lexical similarity (as well as the frame constraint) for scoring. This separation avoids the difficulty of defining the exact interaction between the two. Specifically, we obtain an initial clustering by grouping together all instances which occur in the same syntactic position within a linking—that is, all pairs (v_i, v_j) for which $\phi_{syn}(v_i, v_j) = 1$ are grouped into the same cluster, assuming that arguments occurring in a specific syntactic position under a specific linking share the same role.

We specify the syntactic position of an argument using four cues: the verb's voice (active/passive), the argument's linear position relative to the predicate (left/right), the syntactic relation of the argument to its governor (e.g., subject or object), and the preposition used for realizing the argument (if any). Each argument is assigned a four-tuple consisting of these cues and two syntactic positions are assumed equal iff they agree on all cues.

Whereas the similarity functions defined in Section 3.1 measure role-semantic similarity *between instances* on a particular feature, the scoring function measures role-semantic similarity *between clusters*. Naturally, the similarity between two clusters is defined in terms of the similarities of the instances contained in the clusters. This involves two aggregation stages. Initially, instance similarities are aggregated in each feature layer, resulting in an aggregate score for each feature. These layer-specific scores are then integrated into a single score, which quantifies the overall similarity between the two clusters (see Figure 1).

An obvious way to determine the similarity between two clusters (with respect to a particular feature f) would be to analyze their connectivity. For example, we could use edge density (Schaeffer 2007) to average over the weights of edges between two clusters. However, edge density is an inappropriate measure of similarity in our case, because we cannot assume that arbitrary pairs of instances are similar with respect to a particular feature, even if two clusters represent the same semantic role. Consider for example lexical similarity: Most head words will not agree (even within a cluster) and therefore averaging between all pairs would yield low scores, regardless of whether the clusters represent the same role or not. Analogously, the vast majority of instance pairs from any two clusters will belong to different frames, and thus averaging over all possible pairs of instances would not yield indicative scores.

We therefore adopt an averaging procedure which finds, for each instance in one cluster, the instance in the other cluster that is maximally similar or dissimilar and averages over the scores of these alignments:

$$s_f(c_k, c_l) = \frac{1}{N_k + N_l}\left(\sum_{v_i \in c_k} \operatorname*{abs\,max}_{v_j \in c_l} \phi_f(v_i, v_j) + \sum_{v_j \in c_l} \operatorname*{abs\,max}_{v_i \in c_k} \phi_f(v_i, v_j)\right) \tag{6}$$

Here, abs max is a functional that returns the extreme value of its argument, either positive or negative: $\operatorname{abs\,max}_{x \in X} g(x) = g(\arg\max_{x \in X} |g(x)|)$. Note that the alignments are unconstrained in the sense that $v_a \in c_k$ can be aligned to $v_b \in c_l$ in the first term of Equation (6), while v_b can be aligned to some other instance in the second term. Moreover, alignments in each term are many-to-one, namely, multiple instances from c_k can be aligned to the same $v_b \in c_l$ in the first term and likewise in the second term. This means that score aggregation does not reflect the distributional properties of clusters (e.g., the frequency of head words in each cluster). Consider for example two clusters with an identical set of head words. Because many-to-one alignments are allowed, each instance can be aligned with maximal score to some other instance regardless of the frequencies of these words.

As an alternative, we also use the well-known cosine similarity function—although only for the features based on argument head words (*lex*) and parts of speech (*pos*):

$$s_f(c_k, c_l) = \frac{x_k^f \cdot x_l^f}{\|x_k^f\| \|x_l^f\|}. \tag{7}$$

Here x_k^f and x_l^f are vector representations of the cluster containing as components the occurrence frequencies of a particular value of the feature f (i.e., *lex* and *pos* in our case). Another solution would be to enforce one-to-one alignments and redefine Equation (6) as the optimal bipartite matching between the two clusters. Although this solution adheres to the graph formulation (in contrast to Equation (7)) we see no theoretical reason that makes it superior to cosine similarity. Moreover, its computation would require cubic runtime in the number of vertices using the Hungarian algorithm (Munkres 1957), which is prohibitively slow for sufficiently large clusters.

Layer-specific similarity scores must be combined into an overall cluster similarity score. Because similarity scores and their aggregates for different features are not directly comparable, their combination through summation would require weighting each layer score according to its relative strength. Due to the difficulty of specifying these weights without access to labeled training data, we propose an alternative scheme

that is based on the distinction between positive and negative evidence. Negative evidence is used to rule out a merge, whereas positive evidence provided by the lexical score is used to score merges that have not yet been ruled out:

$$s(c_k, c_l) = \begin{cases} -1 & \text{if } s_{frame}(c_k, c_l) < \alpha \\ -1 & \text{if } s_{pos}(c_k, c_l) < \beta \\ s_{lex}(c_k, c_l) & \text{if } s_{lex}(c_k, c_l) > \gamma \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

When the part-of-speech similarity is below a certain threshold β, or when clause-level constraints are satisfied to a lesser extent than threshold α, the score takes value -1 and the merge is ruled out. If the merge is not ruled out, the lexical similarity score determines the magnitude of the overall score, provided that it is above threshold γ. Otherwise, the function returns 0, indicating that neither strong positive nor negative evidence is available. The cluster-similarity scoring function can be viewed as the decision function of a binary classifier for deciding on whether to merge a particular pair of clusters. The classifier is informed by the similarity scores for each feature layer and outputs a confidence-weighted decision (positive/negative), where the sign $\text{sgn}(\phi_f(v_i, v_j))$ indicates the decision and the absolute value $|\phi_f(v_i, v_j)|$ quantifies confidence. The scoring function in Equation (8) essentially implements a simple decision list classifier, whose decision rules are sequentially inspected from top to bottom, applying the first matching rule.

Although our definition avoids weighting, it has introduced threshold parameters α, β, and γ that we need to somehow estimate. We propose a scheme in which parameters β and γ are iteratively adjusted, and α, the threshold determining the extent to which the frame constraints can be violated, is kept fixed. We heuristically set α to -0.05, based on the intuition that in principle frame constraints must be satisfied although in practice, due to noise we expect a small number of violations (i.e., at most 5% of instances can violate the constraint). Parameters β and γ are initially set to their maximal value 1, thereby ruling out all merges except those with maximal confidence. The parameters then decrease iteratively according to a routine whose pseudo-code is specified in Algorithm 2. The parameter β decreases at each iteration by a small amount (0.025) until it reaches $\epsilon = 0.025$, at which point its value is reset to 1.0 and γ is discounted by a factor close to one (0.9). This is repeated until γ falls below ϵ, upon which the algorithm terminates.

Algorithm 2 Routine for updating the threshold parameters α, β, and γ; it is called after every iteration of Algorithm 1.

```
β ← β − 0.025
if β ≤ 0.0 then
    β ← 1.0
    γ ← 0.9γ
    if γ < ε then
        done ← true
    end
end
```

Runtime Analysis. As described in the previous section, Algorithm 1 stops when the threshold γ falls below some small value ϵ. Both γ and α iteratively decrease based on a fixed scheme. The outer loop and starting in line 1 is therefore computed in constant time T. Each pass through the inner loop starting at line 4 iterates over $O(|C|)$ clusters and for each one of them a score with $O(|C|)$ other clusters is computed. Assume that f_i denotes the fraction of all V instances in cluster c_i, namely, $f_i V = |c_i|$ and $\sum_{i=1}^{|C|} f_i = 1$. Then, overall, the number of instance-wise similarities we need to evaluate is at most $O(|V|^2)$:

$$\sum_{i=1}^{|C|}\sum_{j=i+1}^{|C|}(f_i|V|)(f_j|V|) = \frac{1}{2}\sum_{i=1}^{|C|}\sum_{j=1}^{|C|}(f_i|V|)(f_j|V|) - \frac{1}{2}\sum_{i=1}^{|C|}(f_i|V|)^2$$

$$\leq |V|^2 \sum_{i=1}^{|C|}\sum_{j=1}^{|C|} f_i f_j = |V|^2 \sum_{i=1}^{|C|} f_i \sum_{j=1}^{|C|} f_j = |V|^2$$

The total runtime in terms of the input is therefore $O(T \cdot |V|^2)$. Although this could be prohibitively inefficient for large data sets, we did not observe long runtimes in our experiments. Various optimizations are conceivable—for example, the cluster similarity scores in line 5 of Algorithm 1 can be cached such that they only need to be recomputed when a cluster changes (i.e., it is merged with another cluster).

4.2 Multi-Layer Label Propagation

Our second graph partitioning algorithm is based on the idea of propagating cluster membership information along the edges of a graph, subsequently referred to as **propagation graph**. As we explain in more detail subsequently, compared with agglomerative clustering, this algorithm in principle is less prone to making false greedy decisions that cannot be later revoked. Moreover, it has lower runtime and thus scales better to larger data sets.

The propagation graph is created by collapsing vertices of the initial multi-layer graph. Vertices in the propagation graph represent an *atomic* set of instances of the original graph, that is, a group of instances that are always assigned to the same cluster. For our induction problem, the vertices of the propagation graph correspond to the initial clusters of the agglomerative algorithm discussed in Section 4.1. More formally, let $a_i \in A$ denote the i-th vertex of the propagation graph, which references an atomic cluster of vertices $\{v_{i_1} \ldots v_{i_{N_i}}\}$ of the original graph that occur in the same syntactic position within the same linking. Because each vertex of the propagation graph corresponds to a cluster of vertices in the original graph, the edges of the propagation graph can be defined in terms of the edges between these vertices in the original graph. We reuse Equations (6) and (7) to define the edge weights of the propagation graph as aggregates over the edge weights in the original graph. For each feature layer we define the set of edges as:

$$B_f = \{(a_i, a_j) \in A \times A | s_f(a_i, a_j) \neq 0\} \tag{9}$$

Each edge $(a_i, a_j) \in B_f$ in layer f is accordingly weighted by $s_f(a_i, a_j)$. Each vertex a_i is associated with a label l_i, indicating the partition that a_i and all the vertices in the original graph that have been collapsed into a_i belongs to.

Note that the label propagation algorithm is informed by the same similarity functions as agglomerative clustering and uses an identical initialization procedure but provides an alternative means of *cluster inference*. Initially, each vertex of the propagation graph belongs to its own cluster, that is, we let the number of clusters $L = |A|$ and set $l_i \leftarrow i$. Given this initial vertex labeling, the algorithm proceeds by iteratively updating the label for each vertex (lines 4–10 in Algorithm 3). This crucially relies on a scoring procedure in which a score $s(l)$ is computed for each possible label l. We discuss the details of the scoring procedure below.

The label scoring procedure required in line 5 of Algorithm 3 has parallels to the cluster pair scoring procedure of the agglomerative algorithm. It also consists of two stages: Initially, evidence is collected independently on each feature layer by computing label score aggregates with respect to each feature and then these feature scores are combined in order to arrive at an overall score.

Assume we are updating vertex a_i. The first step is to compute the score for each feature f and each label l:

$$s_f(l) = \sum_{a_j \in \mathcal{N}_i(l)} s_f(a_i, a_j) \tag{10}$$

where $\mathcal{N}_i(l) = \{a_j | (a_i, a_j) \in B_f \wedge l = l_j \wedge |a_j| > |a_i|\}$ denotes the set of a_i's neighbors with label l that are larger than a_i. Intuitively, each neighboring vertex votes for the cluster it is currently assigned to, where the strength of the vote is determined by the similarity to the vertex (i.e., edge weight) being updated. The votes of all (larger) neighboring vertices are counted together, resulting in a score for each possible label. The condition of including only larger vertices for computing the score is analogous to the prioritization mechanism of the agglomerative algorithm (only merges with larger clusters are considered for a given candidate cluster). We impose this restriction for the same reason, namely, that scores for larger clusters are more reliable.

Given the scores $s_f(l)$ for a particular label l on each layer f, our goal then is to combine them into a single overall score $s(l)$ for the label. As in agglomerative partitioning, combining these scores through summation is not possible without "guessing" their weights, and therefore we use a sequential combination instead:

$$s(l) = \begin{cases} -1 & \text{if } s_{frame}(l) < \alpha \\ -1 & \text{if } s_{pos}(l) < \beta \\ s_{lex}(l) & \text{if } s_{lex}(l) > \gamma \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Analogously to Equation (8), negative evidence that stems from part-of-speech information or frame constraints can veto a propagation, whereas positive evidence stemming from argument head words can promote a propagation. If neither strong evidence (positive or negative) is available, the label is assigned a zero score. Note that the scoring function has three parameters with an identical interpretation to those in the scoring function of the agglomerative algorithm. The threshold update that takes place in line 11 of Algorithm 3 is therefore the same as the one described in Section 4.1 for the agglomerative algorithm.

We now analyze the runtime of our algorithm. Let T denote the number of iterations of the outer loop starting at line 1 of Algorithm 3. The inner loop starting at line 4 iterates over $|A|$ clusters and for each one of them it has to evaluate at most $|A|$ neighboring

Algorithm 3 Multi-layer label propagation algorithm.

```
while not done do
    A ← a list of all propagation graph vertices sorted by size (number of
    contained instances) in descending order
    i ← 1
    while i < length(A) do
        l* ← arg max_{l∈{0...L}} s(l)
        if s(l*) > 0 then
            l_i ← l*
        end
        i ← i + 1
    end
    adjust thresholds
end
```

nodes. Additionally, there are the one-time costs of computing the similarities between atomic clusters which take $O(|V|^2)$ time. The total runtime is therefore $O(T|A|^2 + |V|^2)$. Because $|A|^2 << |V|^2$, label propagation is substantially faster than agglomerative clustering.

4.3 Relationship to Single-Layer Graph Partitioning

Clustering algorithms typically assume instance-wise similarities as input (i.e., single-layer graphs). For our role induction problem, this would require a heuristically defined similarity function that combines the similarities on individual features into a single similarity score between instances. In other words, we would collapse the multiple graph layers into a single layer and then partition the resulting single-layer graph according to a standard clustering algorithm. A main difference between the two approaches is the order in which similarities are aggregated: Whereas multi-layer graph partitioning aggregates similarities on each feature layer first and then combines them into an overall *cluster-wise* similarity score, in the single-layer case feature similarities are eagerly combined into an overall *instance-wise* similarity score and then aggregated. Thus, in the multi-layer setting, aggregation can be done in a principled way by considering the individual feature layers in isolation. For large clusters the resulting scores for each feature layer will provide reliable evidence for or against a merge. Combining these cluster-wise similarity scores is much less error-prone than the eager combination at the instance-level used by the single-layer approach. We experimentally confirm this intuition (see Section 5.5) by comparing against the single-layer partitioning algorithm presented in Lang and Lapata (2011b).

5. Role Induction Experiments on English

We adopt the general 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. Our approach is not tied to a specific syntactic representation—both constituent- and dependency-based representations can be used. The bulk of our experiments focus on English data and a dependency-based representation that simplifies argument identification considerably and is consistent with the CoNLL 2008 benchmark data set used for evaluation in our experiments. To show that our method can be applied to other languages and across varying syntactic representations, we also report experiments on German using a constituent-based representation (see Section 6).

Given the parse of a sentence, our system identifies argument instances and assigns them to clusters. Thereafter, argument instances can be labeled with an identifier corresponding to the cluster they have been assigned to, similar to PropBank core labels (e.g., A0, A1). We view argument identification as a syntactic processing step that can be largely undertaken deterministically through analysis of the syntactic tree. We therefore use a small set of rules to detect arguments with high precision and recall. In the following, we first describe the data set (Section 5.1) on which our experiments were carried out. Next, we present the argument identification component of our system (Section 5.2) and the method used for comparison with our approach. Finally, we explain how system output was evaluated (Section 5.4).

5.1 Data

For evaluation purposes, we ran our method on the CoNLL 2008 shared task data set (Surdeanu et al. 2008), which provides PropBank style gold standard annotations. As our algorithm induces *verb-specific* roles, PropBank annotations are a natural choice of gold standard for our problem. The data set contains annotations for verbal and nominal predicate-argument constructions, but we only considered the former. The CoNLL data set was taken from the *Wall Street Journal* portion of the Penn Treebank and converted into a dependency format (Surdeanu et al. 2008). Input sentences are represented in the dependency syntax specified by the CoNLL 2008 shared task (see Figure 2 for an example). In addition to gold standard dependency parses, the data set also contains automatic parses obtained from the MaltParser (Nivre et al. 2007), which we will use as an alternative in our experiments in order to assess the impact of parse quality. For each argument only the head word is annotated with the corresponding semantic role, rather than the whole constituent. We assume that argument heads are content words (e.g., the head of a prepositional phrase is the nominal head rather than the preposition). We do not treat split arguments or co-referential arguments (e.g., in relative clauses). Specifically, we ignore arguments with roles preceded by the C- or R- prefix in the gold standard. All argument lemmas were normalized to lower case; we also replaced numerical quantities with a placeholder; to further reduce data sparsity, we identified

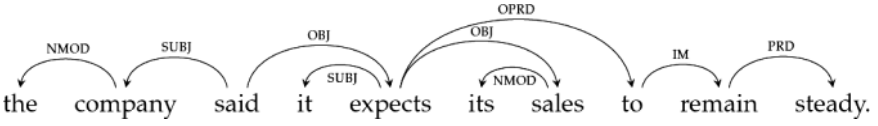

Figure 2
A sample dependency parse with dependency labels SBJ (subject), OBJ (object), NMOD (nominal modifier), OPRD (object predicative complement), PRD (predicative complement), and IM (infinitive marker).

Table 1
Argument identification rules for English.

1.	Discard a candidate if it is a coordinating conjunction or punctuation.
2.	Discard a candidate if the path of relations from predicate to candidate ends with coordination, subordination, etc. (see Appendix A for the full list of relations).
3.	Keep a candidate if it is the closest subject (governed by the subject-relation) to the left of a predicate and the relations from predicate p to the governor g of the candidate are all upward-leading (directed as $g \rightarrow p$).
4.	Discard a candidate if the path between the predicate and the candidate, excluding the last relation, contains a subject relation, adjectival modifier relation, etc. (see Appendix A for the full list of relations).
5.	Discard a candidate if it is an auxiliary verb.
6.	Keep a candidate if it is directly connected to the predicate.
7.	Keep a candidate if the path from predicate to candidate leads along several verbal nodes (verb chain) and ends with an arbitrary relation.
8.	Discard all remaining candidates.

the head of proper noun phrases heuristically as the most frequent lemma contained in the phrase.

5.2 Argument Identification

In the supervised setting, a classifier is used in order to decide for each node in the parse tree whether it represents a semantic argument or not. 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, this decision is deferred to role induction.[4] We assume here that predicate identification is a precursor to argument identification and can be done relatively straightforwardly based on part-of-speech information.

The rules given in Table 1 are used to discard or select argument candidates for English. They primarily take into account the parts of speech and the syntactic relations encountered when traversing the dependency tree from predicate to argument. A priori, all words in a sentence are considered argument candidates for a given predicate. Then, for each candidate, the rules are inspected sequentially and the first matching rule is applied. We will exemplify how the argument identification component works for the predicate *expect* in the sentence *The company said it expects its sales to remain steady* whose parse tree is shown in Figure 2. Initially, all words except the predicate itself are treated as argument candidates. Then, the rules from Table 1 are applied as follows. Firstly, the words *the* and *to* are discarded based on their part of speech (Rule 1); then, *remain* is discarded because the path ends with the relation IM and *said* is discarded as the

4 A few supervised systems implement a similar definition (Koomen et al. 2005), although in most cases the argument identification component makes a *final* positive or negative decision regarding the status of an argument candidate.

path ends with an upward-leading OBJ relation (Rule 2). Rule 3 matches to *it*, which is therefore added as a candidate. Next, *steady* is discarded because there is a downward-leading OPRD relation along the path and the words *company* and *its* are also discarded because of the OBJ relations along the path (Rule 4). Rule 5 does not apply but the word *sales* is kept as a likely argument (Rule 6). Finally, Rule 7 does not apply, because there are no candidates left.

On the CoNLL 2008 training set, our argument identification rules obtain a precision of 87.0% and a recall of 92.1% on gold standard parses. On automatic parses, precision is 79.3% and recall 84.8%. Here, precision measures the percentage of selected arguments that are actual semantic arguments, and recall measures the percentage of actual arguments that are not filtered out.

Grenager and Manning (2006) also devise rules for argument identification, unfortunately without providing any details on their implementation. More recently, attempts have been made to identify arguments without relying on a treebank-trained parser (Abend and Rappoport 2010b; Abend, Reichart, and Rappoport 2009). The idea is to combine a part-of-speech tagger and an unsupervised parser in order to identify constituents. Likely arguments can be in turn identified based on a set of rules and the degree of collocation with the predicate. Perhaps unsurprisingly, this method does not match the quality of a rule-based component operating over trees produced by a supervised parser.

5.3 Baseline Method for Semantic Role Induction

The linking between semantic roles and syntactic positions is not arbitrary; specific semantic roles tend to map onto specific syntactic positions such as subject or object (Levin and Rappaport 2005; Merlo and Stevenson 2001). We further illustrate this observation in Table 2, which shows how often individual semantic roles map onto certain syntactic positions. The latter are simply defined as the relations governing the argument. The frequencies in the table were obtained from the CoNLL 2008 data set and are aggregates across predicates. As can be seen, semantic roles often approximately correspond to a single syntactic position. For example, A0 is commonly mapped onto subject (SBJ), whereas A1 is often realized as object (OBJ).

This motivates a baseline that directly assigns instances to clusters according to their syntactic position. The pseudo-code is given in Algorithm 4. For each verb we allocate $N = 22$ clusters (the maximal number of gold standard clusters together with a default cluster). Apart from the default cluster, each cluster is associated with a syntactic position and all instances occurring in that position are mapped into the cluster. Despite being relatively simple, this baseline has been previously used as a point of comparison by other unsupervised semantic role labeling systems (Grenager and Manning 2006; Lang and Lapata 2010) and shown difficult to outperform. This is partly due to the fact that almost two thirds of the PropBank arguments are either A0 or A1. Identifying these two roles correctly is therefore the most important distinction to make, and because this can be largely achieved on the basis of the arguments' syntactic position (see Table 2), the baseline yields high scores.

5.4 Evaluation

In this section we describe how we assess the quality of a role induction method that assigns labels to units that have been identified as likely arguments. We also discuss how we measure whether differences in model performance are statistically significant.

Table 2
Contingency table between syntactic position and semantic roles. Only the eight most frequent syntactic positions and their labels are listed (i.e., SBJ (Subject), OBJ (Object), ADV (Adverbial), TMP (Temporal), PMOD (Preposition and its child), OPRD (Object complement), LOC (Location), DIR (Direction)). Counts were obtained from the CoNLL 2008 training data set using gold standard parses. The marginals in the right-most column include all syntactic positions (not only the eight most frequent ones). **Boldface** highlights the most frequent role per syntactic position (e.g., SBJ is frequently A0, OBJ is A1).

	SBJ	OBJ	ADV	TMP	PMOD	OPRD	LOC	DIR	Total
A0	**50,473**	3,350	145	4	2,464	28	12	0	60,398
A1	18,090	**50,986**	3,207	45	4,819	3,489	118	170	83,535
A2	1,344	2,741	**6,413**	74	774	2,440	606	800	19,585
A3	88	254	**1,208**	37	116	114	63	940	3,359
A4	6	20	351	7	79	34	28	**2,089**	2,687
A5	0	0	19	0	1	3	0	**28**	67
AA	**10**	1	0	0	1	0	0	0	13
ADV	7	46	**7,364**	33	55	31	103	2	8,070
CAU	3	6	**215**	14	5	0	8	0	1,178
DIR	0	3	304	2	5	1	19	**639**	1,123
DIS	0	3	**3,326**	47	2	0	15	0	4,823
EXT	1	6	**418**	0	6	3	23	4	621
LOC	18	32	358	15	127	2	**5,076**	9	5,831
MNR	7	54	**2,285**	22	59	36	154	6	6,238
MOD	9	**2,130**	77	22	69	3	6	0	9,030
NEG	0	0	**3,078**	39	0	0	0	0	3,172
PNC	1	11	**458**	4	4	292	8	4	2,231
PRD	0	2	**41**	0	0	11	2	0	66
PRT	0	0	0	0	0	0	0	0	2
REC	0	5	**8**	0	0	0	0	0	14
TMP	14	93	969	**14,465**	141	1	42	15	16,086
Total	70,071	59,744	30,248	14,830	8,730	6,488	6,285	4,706	228,129

Arguments are labeled based on the cluster they have been assigned to, which means that in contrast to the supervised setting we cannot verify the correctness of these labels directly (e.g., by comparing them to the gold standard). Instead, we will look at the induced clusters as a whole and assess their quality in terms of how well they reflect the assumed gold standard. Specifically, 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 cluster purity 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

$$PU = \frac{1}{N} \sum_i \max_j |G_j \cap C_i| \tag{12}$$

Collocation is the inverse of purity (van Rijsbergen 1974) and defined as follows. For each gold role, we determine the cluster with the largest number of instances for

Algorithm 4 Baseline method for semantic role induction.

input : argument instances for a particular verb
output: verb-specific clusters of instances

S ← the N most frequent syntactic positions in the dataset
foreach $s \in S$ **do**
  allocate a cluster c_s for s
end
allocate a default cluster $c_\perp$ for all other positions
foreach *instance* x **do**
  $s_x \leftarrow$ syntactic position of x
  if $s_x \in S$ **then**
    assign instance to cluster c_{s_x}
  end
  else
    assign instance to default cluster $c_\perp$
  end
end
return all clusters

that role (the role's *primary* cluster) and then compute the percentage of instances that belong to the primary cluster for each gold role:

$$CO = \frac{1}{N} \sum_j \max_i |G_j \cap C_i| \tag{13}$$

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:

$$F1 = \frac{2 \cdot CO \cdot PU}{CO + PU} \tag{14}$$

Purity and collocation measure essentially the same data traits as precision and recall, which in the context of clustering are, however, defined on pairs of instances (Manning, Raghavan, and Schütze 2008), which makes them a bit harder to grasp intuitively. We therefore prefer purity and collocation, arguing that these should be assessed in combination or together with F1 because they can be traded off against each other. Purity can be trivially maximized by mapping each instance into its own cluster, and collocation can be trivially maximized by mapping all instances into a single cluster.

Although it is desirable to report performance with a single score such as F1, it is equally important to assess how purity and collocation contribute to this score. In particular, if a hypothetical system were to be used for automatically annotating data, low collocation would result in higher annotation effort and low purity would result in lower data quality. Therefore high purity is imperative for an *effective* system whereas

high collocation contributes to *efficient* data labeling. For assessing our methods we therefore introduce the following terminology. If a model attains higher purity than the baseline, we will say that it is **adequate**, because it induced roles that adequately represent semantic roles. If a model attains higher F1 than the baseline, we will say that it is **non-trivial**, because it strikes a tradeoff between collocation and purity that is non-trivial. Our goal then is to find models that are both adequate and non-trivial.

In order to assess whether differences in performance between two models are statistically significant, we used a sign test. Specifically, we obtained a series of score pairs by testing two methods on a subsample of the test data. Each subsample corresponds to a random selection of $M = 2,000$. We consider the resulting samples to be "sufficiently" independent to obtain indicative results from the test. As null hypothesis (H_0) we assume that a model m attains scores equal to another model b. Under H_0 the probability that model m outperforms model b on a particular test set is $\frac{1}{2}$. The random variable S counting the number of times that $score_m > score_b$ in a sample of N score pairs is binomially distributed:

$$S = \sum_{i=1}^{N} \mathbf{1}[score_m^{(i)} > score_b^{(i)}]\ Bin(\frac{1}{2}, N) \tag{15}$$

We can therefore use S as our test statistic and reject the null hypothesis H_0 if $S >> \frac{N}{2}$.

5.5 Results

Our results are summarized in Tables 3–5, which report cluster purity (PU), collocation (CO), and their harmonic mean (F1) for the baseline and our two multi-layer graph partitioning algorithms. We present scores on four data sets that result from the combination of automatic parses with automatically identified arguments (auto/auto), gold parses with automatic arguments (gold/auto), automatic parses with gold arguments (auto/gold), and gold parses with gold arguments (gold/gold). We show how performance varies for our methods when measuring cluster similarity in the two ways described above: (a) by finding for each instance in one cluster the instance in the other cluster that is maximally similar or dissimilar and averaging over the scores of these alignments (avgmax) and (b) by using cosine similarity (see Section 4.1). We also report results for the single-layer algorithm proposed in Lang and Lapata (2011b).[5] Given a verbal predicate, they construct a single-layer graph whose edge weights express instance-wise similarities directly. 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 among the neighbors of the vertex being updated.

Both agglomerative partitioning and multi-layered label propagation algorithms systematically achieve higher F1 scores than the baseline—that is, induce non-trivial clusterings and more adequate semantic roles (by attaining higher purity). For example, on the auto/auto data set, the agglomerative algorithm using cosine similarity

5 The results in Table 5 differ slightly from those published in Lang and Lapata (2011b). This is due to a small change in the preprocessing of the data. For all English experiments reported here, we removed arguments with R- and C- role prefixes and replaced numbers with a placeholder.

Table 3
Results for agglomerative partitioning (for avgmax and cosine similarity). F1 improvements over the baseline are statistically significant in all settings ($q < 0.001$). **Boldface** highlights the best performing system according to purity, collocation, and F1.

Parse/Arg	Baseline			Agglomerative					
				avgmax			cosine		
	PU	CO	F1	PU	CO	F1	PU	CO	F1
auto/auto	68.3	**72.1**	70.1	75.3	69.2	72.1	**75.5**	69.5	**72.4**
gold/auto	74.9	**78.5**	76.6	80.3	73.8	76.9	**80.7**	74.0	**77.2**
auto/gold	77.0	71.5	74.1	84.9	70.8	77.2	**85.6**	**71.9**	**78.1**
gold/gold	81.6	**78.1**	79.8	87.4	75.3	80.9	**87.9**	75.6	**81.3**

Table 4
Results for multi-layered label propagation (for avgmax and cosine similarity). F1 improvements over the baseline are statistically significant in all settings ($q < 0.001$). **Boldface** highlights the best performing system according to purity, collocation, and F1.

Parse/Arg	Baseline			Multi-Layer Label Propagation					
				avgmax			cosine		
	PU	CO	F1	PU	CO	F1	PU	CO	F1
auto/auto	68.3	**72.1**	70.1	73.8	70.3	72.0	**74.0**	70.3	**72.1**
gold/auto	74.9	**78.5**	76.6	78.8	74.3	76.5	**79.2**	74.3	**76.7**
auto/gold	77.0	71.5	74.1	82.9	72.8	77.5	**83.6**	**73.1**	**78.0**
gold/gold	81.6	**78.1**	79.8	85.6	75.8	80.4	**86.3**	76.1	**80.9**

Table 5
Results for single-layered label propagation using a heuristic similarity function. F1 improvements over the baseline are statistically significant ($q < 0.001$) in the auto/gold and gold/gold settings. **Boldface** highlights the best performing system according to purity, collocation, and F1.

Parse/Arg	Baseline			Label Propagation		
	PU	CO	F1	PU	CO	F1
auto/auto	68.3	**72.1**	70.1	**70.1**	70.4	**70.2**
gold/auto	74.9	**78.5**	76.6	**76.4**	77.2	**76.8**
auto/gold	77.0	71.5	74.1	**79.6**	**72.6**	**75.9**
gold/gold	81.6	78.1	79.8	**83.7**	**78.2**	**80.9**

increases F1 by 2.3 points over the baseline and by 7.2 points in terms of purity. This increase in purity is achieved by trading off against collocation, although in a favorable ratio as indicated by the overall higher F1. All improvements over the baseline are statistically significant ($q < 0.001$ according to the test described in Section 5.4). In general, we observe that cosine similarity outperforms avgmax similarity. We conjecture that cosine is a more appropriate measure of cluster similarity for features where it is beneficial to capture the distributional similarity of clusters. The two algorithms perform comparably—differences in F1 are not statistically significant (except in the gold/auto setting). Nevertheless, agglomerative partitioning obtains higher purity and F1 than label propagation. The latter trades off more purity and in return obtains higher collocation. The single-layer method is inferior to the multi-layer algorithms,

in particular because it is less robust to noise, as demonstrated by the markedly worse results on automatic parses. On the auto/auto data set the single-layered algorithm is on a par with the baseline and marginally outperforms it on the auto/gold and gold/gold configurations.

To help put our results in context, we compare our methods with Titov and Klementiev's (2012a) Bayesian clustering models. They report results on the CoNLL 2008 data sets with two model variants, a factored model that models each verb independently and a coupled model where model parameters are shared across verbs. In an attempt to reduce the sparsity of the argument fillers, they also present variants of the factored and coupled models where the argument heads have been replaced by lexical cluster ids stemming from Brown et al.'s (1992) clustering algorithm on the RCV1 corpus. In Table 6 we follow Titov and Klementiev (2012a) and show results on the gold/gold and gold/auto settings. As can be seen, both the agglomerative clustering and label propagation perform comparably to their coupled model, even though they do not implement any specific mechanism for sharing clustering preferences across verbs. Versions of their models that use Brown word clusters (i.e., Factored+Br and Coupled+Br) yield overall best results. We expect this type of preprocessing to also increase the performance of our models, however we leave this to future work. Finally, we should point out that Titov and Klementiev (2012a) do not cluster adjunct-like modifier arguments that are already explicitly represented in syntax (e.g., TMP, LOC, DIR). Thus, their Coupled+Mods model is most comparable to ours in terms of the clustering objective as it treats both core and adjunct arguments and does not make use of the Brown clustering. Table 6 shows the performance of Coupled+Mods on the gold/gold setting only because auto/gold results are not reported.

We further examined the output of the baseline and our best performing model in order to better understand where the performance gains are coming from. Table 7 shows how the two approaches differ when it comes to individual roles. We observe that the agglomerative clustering algorithm performs better than the baseline on all core roles. There are some adjunct roles for which the baseline obtains a higher F1. This is not surprising because the parser directly outputs certain labels such as LOC and TMP which results in high baseline scores for these roles. A word of caution is necessary here since core roles are defined individually for each verb and need not have a uniform corpus-wide interpretation. Thus, conflating per-role scores across verbs is only meaningful to the extent that these labels actually signify the same role (which is mostly true for A0 and A1). Furthermore, the purity scores we provide in this context are averages over the clusters for which the specified role is the majority role.

Table 6
Semantic role induction with graph partitioning and Bayesian clustering.

Model	gold/gold			auto/gold		
	PU	CO	F1	PU	CO	F1
Baseline	81.6	78.1	79.8	77.0	71.5	74.1
Agglomerative	87.9	75.6	81.3	85.6	71.9	78.1
Multi-Layer LP	86.3	76.1	80.9	83.6	73.1	78.0
Factored	88.1	77.1	82.2	85.1	71.8	77.9
Coupled	89.3	76.6	82.5	86.7	71.2	78.2
Coupled+Mods	89.2	74.0	80.9	—	—	—
Factored+Br	86.8	78.8	82.6	83.8	74.1	78.6
Coupled+Br	88.7	78.1	83.0	86.2	72.7	78.8

Table 7
Results for individual roles on the auto/auto data set; comparison between the baseline and the agglomerative clustering algorithm with the cosine similarity function. **Boldface** highlights the best performing system according to purity, collocation, and F1.

Role	Freq	Baseline			Agglomerative		
		PU	CO	F1	PU	CO	F1
A0	49,956	68.2	89.6	77.5	**71.1**	**90.0**	**79.4**
A1	72,032	77.5	75.2	76.3	**80.7**	**76.9**	**78.7**
A2	16,795	65.7	**71.4**	68.4	**79.1**	68.3	**73.3**
A3	2,860	45.4	**81.8**	58.4	**71.7**	80.1	**75.7**
A4	2,471	61.6	**86.1**	71.8	**81.6**	85.1	**83.3**
A5	44	46.4	59.1	52.0	**92.5**	**84.1**	**88.1**
AA	9	46.7	**100.0**	63.6	**50.0**	**100.0**	**66.7**
ADV	5,824	33.8	**86.3**	48.6	**67.7**	41.9	**51.8**
CAU	878	67.5	**79.3**	72.9	**81.5**	73.9	**77.5**
DIR	811	51.5	**71.6**	59.9	**66.9**	58.9	**62.7**
DIS	3,022	36.1	**90.4**	51.6	**57.5**	75.7	**65.3**
EXT	536	46.9	91.0	61.9	**70.2**	**92.2**	**79.7**
LOC	4,481	65.1	**76.5**	**70.4**	**74.2**	58.4	65.3
MNR	5,066	62.0	**64.6**	**63.3**	**84.3**	48.3	61.5
MOD	8,064	80.2	44.1	56.9	**90.3**	**89.3**	**89.8**
NEG	2,952	38.7	98.6	55.6	**53.5**	**98.7**	**69.4**
PNC	1,682	67.9	**71.8**	69.8	**77.8**	70.6	**74.1**
PRD	56	39.1	**92.9**	55.1	**80.4**	85.7	**83.0**
REC	9	25.0	**100.0**	40.0	**75.0**	**100.0**	**85.7**
TMP	12,928	71.1	**78.7**	**74.7**	**73.1**	43.1	54.2
NONE	49,663	57.1	**47.3**	51.8	**71.6**	44.8	**55.1**

We further investigated the degree to which the baseline and the agglomerative clustering algorithm agree in their role assignments. The overall mean overlap was 46.03%. Figure 3a shows the percentage of verbs for which the baseline and our algorithm have no, some, or complete overlap. We discretized overlap into 10 bins of equal size ranging from 0 to 100. We observe that the role assignments produced by the two methods have nothing in common for approximately 13.6% verbs, whereas assignments are identical for 18.1% verbs. Aside from these two bins (see 0 and 100 in Figure 3), a large number of verbs seems to exhibit overlap in the range of 40–60%. Figure 3b shows how the overlap in the cluster assignments varies with verb frequency. Perhaps unsurprisingly, we can see that overlap is higher for least frequent and therefore less ambiguous verbs. In general, although the two methods have some degree of overlap, agglomerative clustering does indeed manage to change and improve the original role assignments of the baseline.

An interesting question concerns precisely the type of changes affected by the agglomerative clustering algorithm over the assignments of the baseline. To be able to characterize these changes we first examined the consistency of the role assignments created by the two algorithms. Specifically, we would expect a verb-argument pair to be mostly assigned to the same cluster (i.e., an argument to bear the same role label for the same verb). Of course this is not a hard constraint as arguments and predicates can be ambiguous and their roles may vary in specific syntactic configurations and contexts. To give an idea of an upper bound, in our gold standard, an argument instance of the same verb bears on average 2.23 distinct roles. For comparison, the baseline creates (on average) 2.9 role clusters for an argument, whereas agglomerative clustering yields more consistent assignments, with an average of 2.34 role clusters per argument.

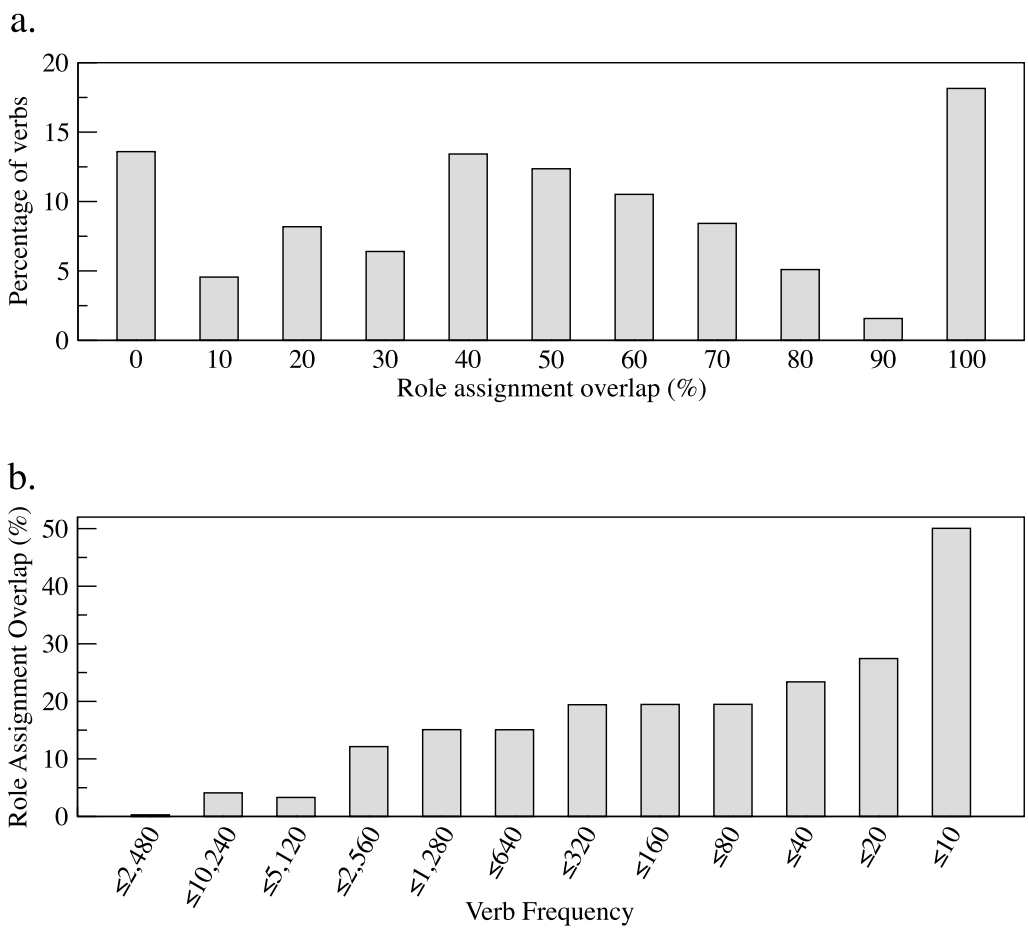

Figure 3
Role assignment overlap between the baseline and agglomerative clustering on the auto/auto data set. Figure 3a shows the percentage of verbs with no overlap (0%), 10% overlap, 20% overlap, 30% overlap, and so on. Figure 3b shows how role overlap varies with verb frequency. Results are reported on the auto/auto data set.

We further grouped the verbs in our data set into different bins according to their polysemy and allowable argument realizations. Specifically, we followed Levin's (1993) taxonomy and grouped verbs according to the number of semantic classes they inhabit (e.g., one, two, and so on). We also binned verbs according to the number of alternations they exhibit. To give an example, the verb *donate* is a member of the CONTRIBUTE class and participates in the causative/inchoative and dative alternations, whereas the verb *shower* is a member of four classes (i.e., SPRAY/LOAD, PELT, DRESS, and WEATHER) and participates in the understood reflexive object and spray/load alternations. Figures 4a,b show the overlap in role assignments between the baseline and agglomerative clustering and how it varies according to verb class ambiguity and argument structure; figures 4c,d illustrate the same for role assignments and their consistency. As can be seen, there is less overlap between the two methods when the verbs in question are more polysemous (Figures 4a) or exhibit more variation in their argument structure (Figure 4b). As far as consistency in role assignments is concerned, agglomerative clustering appears overall more consistent than the baseline. As expected, the mean

role assignment is slightly higher for polysemous verbs because differences in meaning manifest themselves in different argument realizations.

Figure 5 shows how purity, collocation, and F1 vary across alternations and verb classes. Perhaps unsurprisingly, performance is generally better for least ambiguous verbs exhibiting a small number of alternations. In general, agglomerative clustering achieves higher purity across the board whereas the baseline achieves higher collocation. Although agglomerative clustering achieves a consistently higher F1 over the baseline, the performance of the two algorithms converges for the most polysemous verbs (i.e., those inhabiting more than six semantic classes; see Figure 5f). Interestingly, also note that F1 is comparable for verbs with less varied argument structure (i.e., verbs inhabiting one alternation; see Figure 5c). For such verbs the performance gap between the baseline and the agglomerative algorithm is narrower both in terms of purity and collocation. Overall, we observe that agglomerative clustering is able to change some of the role assignments of the baseline for verbs exhibiting a good degree of alternations and polysemy.

Table 8 reports results for 12 individual verbs for the best performing method (i.e., agglomerative partitioning using cosine similarity) on the auto/auto data set. These verbs were selected so as to exhibit varied occurrence frequencies and alternation patterns. As can be seen, the macroscopic result—higher F1 due to significantly higher purity—seems to consistently hold also across verbs. An important exception is the verb

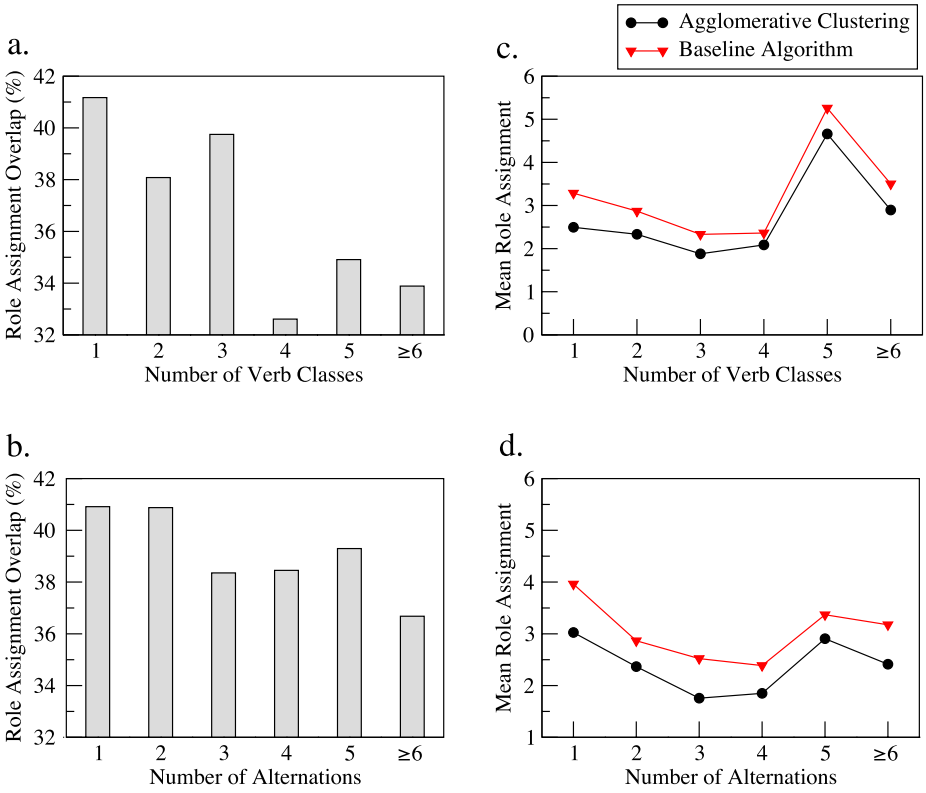

Figure 4
Comparison between the baseline and the agglomerative clustering algorithm in terms of role assignment overlap (a and b) and consistency (c and d). Verbs are grouped according to polysemy (a and c) and number of alternations (b and d). All results are reported on the auto/auto data set.

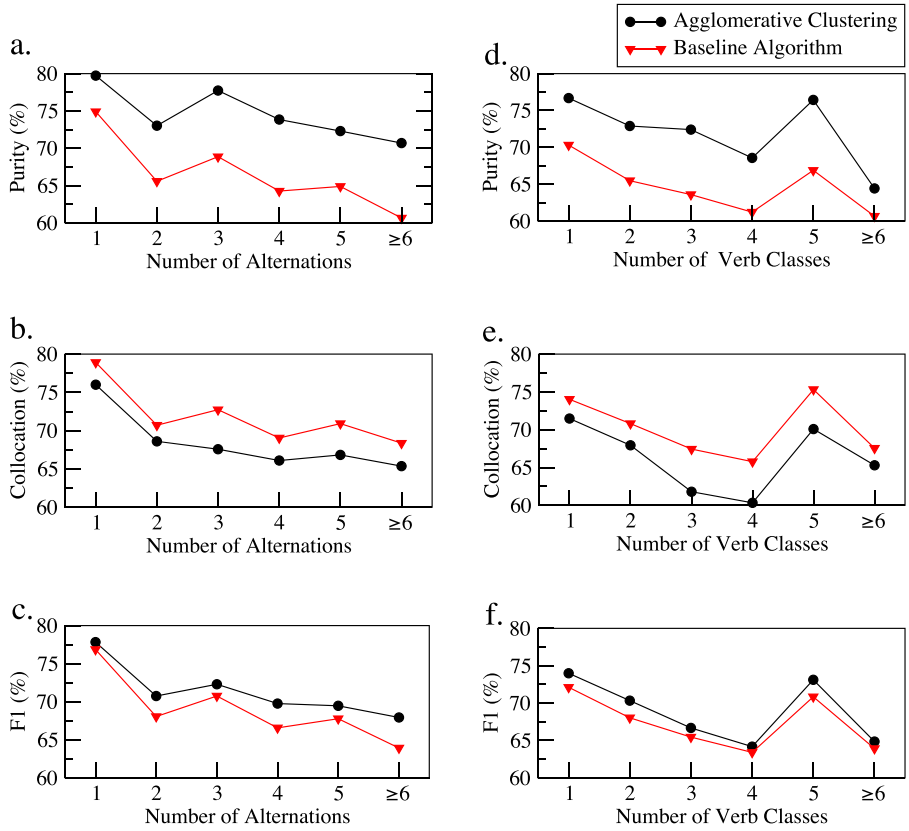

Figure 5
Comparison between the baseline and the agglomerative clustering algorithm across alternations (a–c) and verb classes (d–f) using purity, collocation, and F1. All results are reported on the auto/auto data set.

say, for which the baseline attains high scores due to little variation in its syntactic realization within the corpus. Example output is given in Table 9, which shows the five largest clusters produced by the baseline and agglomerative partitioning for the verb *increase*. For each cluster we list the 10 most frequent argument head lemmas. In this case, our method managed to induce an A0 cluster that is not present in the top five clusters of the baseline, although the cluster also incorrectly contains some A1 arguments that stem from a false merge.

6. Role Induction Experiments on German

The applicability of our method to arbitrary languages is important from a theoretical and practical perspective. On the one hand, linguistic theory calls for models which are universal and generalize across languages. This is especially true for models operating on the (frame-) semantic level, which is a generalization over surface structure and should therefore be less language specific (Boas 2005). On the other hand, a language-independent model can be applied to arbitrary languages, genres, and domains and is thus of greater practical benefit. Because our approach is based on the language-independent principles discussed in Section 1, we argue that it can easily generalize to other languages. To test this claim, we further applied our methods to German data.

Table 8
Results for individual verbs on the auto/auto data set; comparison between the baseline and our agglomerative clustering algorithm with the cosine similarity function. **Boldface** highlights the best performing system according to purity, collocation, and F1.

Verb	Freq	Baseline			Agglomerative		
		PU	CO	F1	PU	CO	F1
say	16,698	**86.7**	**90.8**	**88.7**	85.8	90.4	88.0
make	4,589	63.3	**71.0**	67.0	**66.4**	**71.0**	**68.6**
go	2,331	47.3	**56.0**	51.3	**55.7**	55.3	**55.5**
increase	1,425	58.0	69.0	63.0	**59.2**	**71.5**	**64.8**
know	1,083	58.3	**70.8**	**63.9**	**58.6**	62.0	60.2
tell	969	59.0	**76.8**	66.7	**71.4**	68.0	**69.7**
consider	799	60.7	**65.3**	62.9	**71.0**	60.2	**65.1**
acquire	761	70.7	**78.4**	74.4	**72.0**	77.8	**74.8**
meet	616	70.0	**72.2**	71.1	**78.9**	68.3	**73.2**
send	515	68.3	**67.4**	67.9	**75.9**	64.9	**70.0**
open	528	55.3	**67.8**	**60.9**	**61.9**	55.1	58.3
break	274	51.1	**59.1**	54.8	**62.8**	55.8	**59.1**

Table 9
Five largest clusters created by the baseline and agglomerative partitioning for the verb *increase*. Symbols $ and CD are used as placeholders for monetary units and cardinal numbers, respectively.

Role	Baseline
A0	it, sales, revenue, company, profit, rates, they, earnings, we, number
A1	number, reserves, stake, sales, costs, will, board, demand, rates, capacity
A4	$, %, CD, yen, cent, #, member, earlier, kronor, years
ADV	$, not, CD, also, be, increase, greatly, month, %, thus
A2	%, $, CD, average, significantly, penny, yen, days, slightly, share

Role	Agglomerative
A1	%, number, costs, sales, reserves, demand, stake, competition, pressure, size
A0	it, sales, revenue, company, profit, rates, earnings, we, they, line
A4	$, %, CD, yen, cent, member, result, #, kronor, barrels
A3	$, CD, %, yen, cent, earlier, period, #, member, quarter
TMP	year, quarter, month, years, period, september, CD, week, example, instance

Although on a high-level, German clauses do not differ drastically from English ones with respect to their frame-semantic make-up, there are differences in terms of how frame elements are mapped onto specific positions on the linear surface structure of a sentence, beyond any variations observed among English verbs. In general, German places fewer constraints on word order (more precisely phrase order) and instead relies on richer morphology to help disambiguate the grammatical functions of linguistic units. In particular, verbal nominal arguments are marked with a grammatical case[6] that directly indicates their grammatical function. Although in main declarative clauses the inflected part of the verb has to occur in second position, German is commonly

6 German has (partially ambiguous) markers for Nominative, Accusative, Dative, and Genitive.

considered a verb-final language. This is because the verb often takes the final position in subordinate clauses, as do infinitive verbs (Brigitta 1996).

6.1 Data

We conducted our experiments on the SALSA corpus (Burchardt et al. 2006), a lexical resource for German, which, like FrameNet for English, associates predicates with frames. SALSA is built as an extra annotation layer over the TIGER corpus (Brants et al. 2002), a treebank for German consisting of approximately 40,000 sentences (700,000 tokens) of newspaper text taken from the *Frankfurter Rundschau*, although to date not all predicate-argument structures have been annotated. The frame and role inventory of SALSA was taken from FrameNet, but has been extended and adapted where necessary due to lack of coverage and cross-lingual divergences.

The syntactic structure of a sentence is represented through a constituent tree whose terminal nodes are tokens and non-terminal nodes are phrases (see Figure 6). In addition to labeling each node with a constituent type such as *Sentence*, *Noun Phrase*, and *Verb Phrase*, the edges between a parent and a child node are labeled according to the function of the child within the parent constituent, for example, *Accusative Object*, *Noun Kernel*, or *Head*. Edges can cross, allowing local and non-local dependencies to be encoded in a uniform way and eliminating the need for traces. This approach has significant advantages for non-configurational languages such as German, which exhibit a rich inventory of discontinuous constituents and considerable freedom with respect to word order (Smith 2003). Compared with the Penn TreeBank (Marcus, Santorini, and Marcinkiewicz 1993), tree structures are relatively flat. For example, the tree does not encode whether a constituent is a verbal argument or adjunct; this information is encoded through the edge labels instead.

The frame annotations contained in SALSA do not cover all of the predicate-argument structures of the underlying TIGER corpus. Only a subset of around 550 predicates with approximately 18,000 occurrences in the corpus have been annotated. Moreover, only core roles are annotated, whereas adjunct roles are not, resulting in a smaller number of arguments per predicate (1.96 on average) compared with the CoNLL 2008 data set (2.57 on average) described in section 5.1. Because our

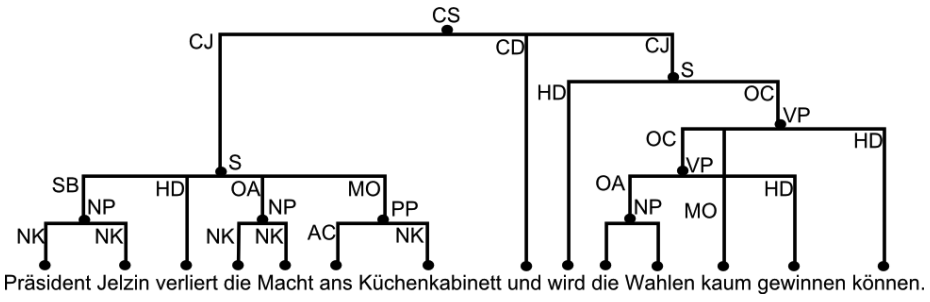

Figure 6
A sample parse tree for the sentence *Präsident Jelzin verliert die Mach ans Küchenkabinett und wird die Wahlen kaum gewinnen können* [translated in English as *President Jelzin loses power to the kitchen cabinet and will hardly be able to win the elections*]. The parse tree contains phrase labels NP (Noun Phrase), PP (Prepositional Phrase), VP (Verb Phrase), S (Sentence), and CS (Coordinated Sentence). The dependency labels are NK (Noun Kernel), SB (Subject), AO (Object Accusative), HD (Head), MO (Modifier), AC (Adpositional Case Marker), CJ (Conjunct), and OC (Clausal Object).

method is designed to induce verb-specific frames, we converted the SALSA frames into PropBank-like frames by splitting each frame into several verb-specific frames and accordingly mapping frame roles onto verb-specific roles. Our data set is comparable to the German data set released as part of the CoNLL 2009 shared task (Hajič et al. 2009), which was also derived from the SALSA corpus. However, we did not convert the original constituent-based SALSA representation into dependencies, as we wanted to assess whether our methods are also compatible with phrase structure trees.

6.2 Experimental Set-up

Although we follow the same experimental set-up as described in Section 5 for English, there are some deviations due to differences in the data sets utilized for the two languages. Firstly, in contrast to the CoNLL 2008 data set, the SALSA data set (and the underlying TIGER corpus) does not supply automatic parse trees and we therefore conducted our experiments on gold parses only. Moreover, because adjunct arguments are not annotated in SALSA, and because argument identification is not the central issue of this work, we chose to also consider only the gold argument identification. Thus, all our experiments for German were carried out on the gold/gold data set.

A substantial linguistic difference between the German and English data sets is the sparsity of the argument head lemmas, which is significantly higher for German than for English: In the CoNLL 2008 data set, the average number of distinct head lemmas per verb is only 3.69, whereas in the SALSA data set it is 20.12. This is partly due to the fact that the *Wall Street Journal* text underlying the English data is topically more focused than the *Rundschau* newspaper text, which covers a broader range of news beyond economics and politics. Moreover, noun compounding is more commonly used in German than in English (Corston-Oliver and Gamon 2004), which leads to higher lexical sparsity.

Data sparsity affects our method, which crucially relies on lexical similarity for determining the role-equivalence of clusters. Therefore, we reduced the number of syntactic cues used for cluster initialization in order to avoid creating too many small clusters for which similarities cannot be reliably computed. Specifically, only the syntactic position and function word served as cues to initialize our clusters. Note that, as in English, the relatively small number of syntactic cues that determine the syntactic position within a linking is a consequence of the size of our evaluation data set (which is rather small) and not an inherent limitation of our method. On larger data sets, more syntactic cues could and should be incorporated in order to increase performance.

In our experiments we compared the baseline introduced in Section 5.3 against agglomerative partitioning and the label propagation algorithm using both cosine- and avgmax-similarity. The parameters α, β, and γ, which determine the thresholds used in defining overall similarity scores, were set and updated identically as for English (i.e., these parameters can be considered language-independent).

6.3 Results

Table 10 reports results for the baseline and our role induction methods, namely, agglomerative clustering and multi-layered label propagation (using the avgmax and cosine similarity functions) on the SALSA gold/gold data set. For comparison, we also include results on the English CoNLL-2008 gold/gold data set. As can be seen, the baseline obtains a similar F1 for German and English, although the contributions of purity and collocation are different for the two languages. In English, purity is

Table 10
Results of agglomerative partitioning and label propagation for cosine and avgmax similarity on German. For comparison purposes results for English on the gold/gold data set are also tabulated. All improvements over the baseline are statistically significant at significance level $q < 0.001$.

Model	German			English		
	PU	CO	F1	PU	CO	F1
Baseline	75.0	**81.7**	78.2	81.6	**78.1**	79.8
Agglomerative (avgmax)	**77.6**	80.8	**79.2**	87.3	75.3	80.9
Agglomerative (cosine)	**77.6**	80.8	**79.2**	**87.9**	75.6	**81.3**
Label Propagation (avgmax)	77.4	80.9	79.1	85.6	75.8	80.4
Label Propagation (cosine)	77.5	81.0	**79.2**	86.3	76.0	80.9

noticeably higher than in German, whereas collocation is higher in German. This is not surprising when taking into account the distribution of syntactic relations governing an argument. A few frequent relation labels absorb most of the probability mass in German (see Figure 7b), whereas the mass is distributed more evenly among the labels in English (Figure 7a), thus leading to higher purity but lower collocation.

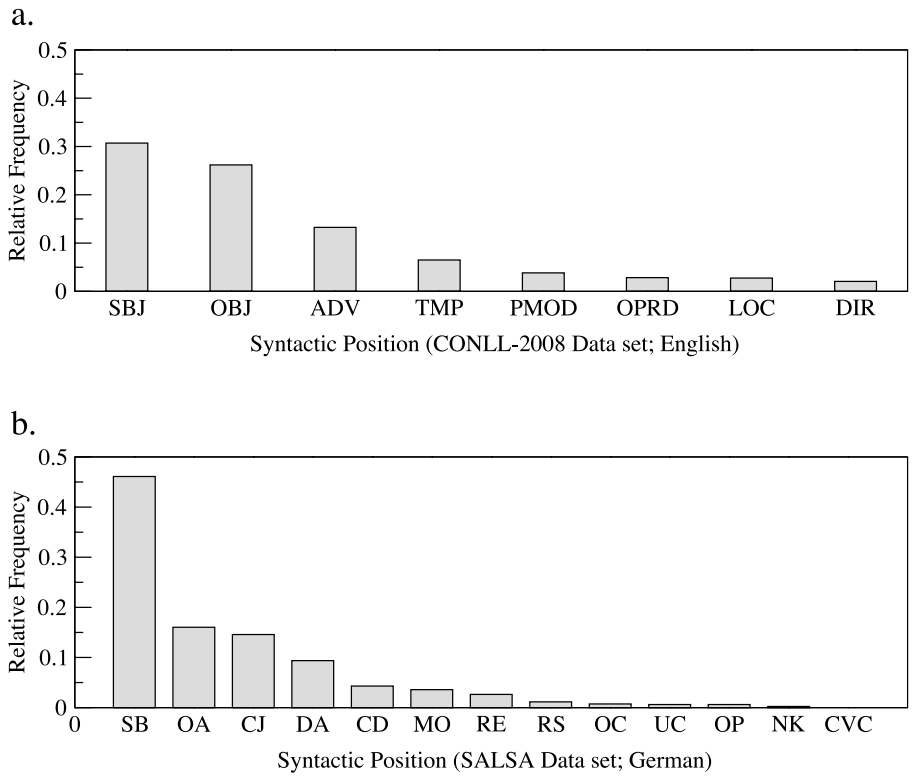

Figure 7
Distribution of syntactic relations governing an argument in English and German data sets. Only the most frequent relations are shown (a key for the English relations is given in Table 2; in German the relations are SB (Subject), OA (Object Accusative), CJ (Conjunct), DA (Dative), CD (Coordinator), MO (Modifier), RE (Subordinate Clause), RS (Reported Speech), OC (Object Clausal), OP (Object Prepositional), NK (Noun Kernel), and CVC (Collocational Verb Construction).

Table 11
Results for individual verbs on the gold/gold SALSA data set; comparison between the baseline and the agglomerative clustering algorithm with the cosine similarity function.

Verb	Freq	Baseline			Agglomerative (cosine)		
		PU	CO	F1	PU	CO	F1
sagen [*say*]	2,076	96.3	89.0	92.5	**97.3**	**97.7**	**97.5**
wissen [*know*]	487	79.7	76.0	77.8	**80.1**	**80.3**	**80.2**
berichten [*report*]	438	79.5	78.3	78.9	**80.0**	**81.3**	**80.7**
nehmen [*take*]	420	49.8	70.2	58.3	**51.9**	**72.4**	**60.5**
verurteilen [*convict*]	265	**70.9**	**83.4**	**76.7**	70.6	81.9	75.8
erhöhen [*increase*]	120	58.3	70.8	64.0	**70.8**	**73.3**	**72.1**
schließen [*close*]	93	40.9	72.0	52.1	**53.8**	**78.5**	**63.8**
brechen [*break*]	45	40.0	**91.1**	55.6	**44.4**	**91.1**	**59.7**
schauen [*watch*]	35	82.9	**91.4**	**86.9**	**85.7**	71.4	77.9
plazieren [*place*]	18	55.6	**83.3**	**66.7**	**66.7**	61.1	63.8
treffen [*meet*]	14	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
regnen [*rain*]	12	66.7	**83.3**	**74.1**	**83.3**	50.0	62.5

In German, our role induction algorithms improve over the baseline in terms of F1. All four methods perform comparably and manage to strike a tradeoff between collocation and purity that is non-trivial and represents semantic roles adequately. Compared with English, the difference between the baseline and our algorithms is narrower. This is because we use fewer syntactic cues for initialization in German, due to the increased data sparsity discussed in the previous section. This also explains why there is little variation in the collocation and purity results across methods. However, qualitatively the tradeoff between purity and collocation is the same as for English (i.e., purity is increased at the cost of collocation).

Tables 11 and 12 show per-verb and per-role results, respectively, for agglomerative clustering using cosine similarity. We report per-verb scores for a selection of 10 verbs

Table 12
Results for individual roles on gold/gold SALSA data set; comparison between the baseline and the agglomerative clustering algorithm with the cosine similarity function.

Role	Freq	Baseline			Agglomerative (cosine)		
		PU	CO	F1	PU	CO	F1
Agent	1,908	70.4	92.8	80.1	**70.5**	**93.9**	**80.5**
Theme	1,637	69.1	79.2	73.8	**69.2**	**79.7**	**74.1**
Cognizer	1,244	75.7	94.3	84.0	**76.2**	**94.6**	**84.4**
Entity	1,195	**79.7**	85.9	**82.7**	78.6	**86.7**	82.4
Content	1,136	87.2	65.2	74.6	**88.7**	**66.8**	**76.2**
Goal	1,071	62.0	**81.0**	70.2	**87.0**	67.2	**75.9**
Topic	477	85.2	**69.4**	**76.5**	**86.8**	58.9	70.2
Source	267	**71.6**	**94.0**	**81.3**	66.1	76.0	70.7
Goods	171	73.0	**68.4**	**70.6**	**74.8**	66.7	70.5
Buyer	121	65.0	**90.1**	75.5	**70.4**	88.4	**78.4**
Employee	63	**50.4**	**98.4**	**66.7**	**50.4**	**98.4**	**66.7**
Required Situation	56	**60.3**	78.6	68.3	52.1	**82.1**	**63.8**
Opinion	50	66.7	50.0	57.1	**69.0**	**62.0**	**65.3**
Leader	29	**86.7**	**69.0**	**76.8**	**86.7**	65.5	74.6
Financed	25	79.3	**64.0**	70.8	**80.0**	**64.0**	**71.1**

(see Table 12a), which in some cases are translations of the verbs used for English. With respect to per-role scores, we make use of the fact that roles have a common meaning across predicates (like A0 and A1 in PropBank), and report scores for a selection of 15 different roles (Table 12b) with varied occurrence frequencies. Per-verb results confirm that data sparsity affects performance in German. As can be seen, agglomerative clustering outperforms the baseline on high-frequency verbs that are less affected by sparsity, although this is not always the case on lower-frequency verbs. Analogously, the method tends to perform better on high-frequency roles, whereas there is no clear trend on lower-frequency roles. In contrast to English, for more than half of the verbs the method manages to outperform the baseline in terms of both purity *and* collocation, which is consistent with our macroscopic result, where the tradeoff between purity and collocation is not as strong as for English.

The experiments show that our methods can be successfully applied to languages other than English, thereby supporting the claim that they are based on a set of language-independent assumptions and principles. Despite substantial differences between German and English grammar, both generally and in terms of the specific syntactic representation that was used, our methods increased F1 over the baseline for both languages and resulted in a similar tradeoff between purity and collocation. Improvements were observed in spite of pronounced data sparsity in the case of German. Recall that we had to reduce the number of syntactic initialization cues in order to be able to obtain results on the relatively small amount of gold-standard data. We would also like to note that porting our system to German did not require any additional feature engineering or algorithmic changes.

7. Conclusions

In this article we described an unsupervised method for semantic role induction in which argument-instance graphs are partitioned into clusters representing semantic roles. A major hypothesis underlying our work has been that semantic roles can be induced without human supervision from a corpus of syntactically parsed sentences based on three linguistic principles : (1) arguments in the same syntactic position (within a specific linking) bear the same semantic role, (2) arguments within a clause bear a unique role, and (3) clusters representing the same semantic role should be more or less lexically and distributionally equivalent. Based on these principles we have formulated a *similarity-driven* model and introduced a *multi-layer* graph partitioning approach that represents similarity between clusters on multiple feature layers, whose connectivity can be analyzed separately and then combined into an overall cluster-similarity score.

Our work has challenged the established view that supervised learning is the method of choice for the semantic role labeling task. Although the proposed unsupervised models do yet achieve results comparable to their supervised counterparts, we have been able to show that they consistently outperform the syntactic baseline across several data sets that combine automatic and gold parses, with gold and automatic argument identification in English and German. Our methods obtain F1 scores that are systematically above the baseline and the purity of the induced clusters is considerably higher, although in most cases this increase in purity is achieved by decreasing collocation. In sum, these results provide strong empirical evidence towards the soundness of our method and the principles they are based on.

In terms of modeling, we have contributed to the body of work on similarity-driven models by demonstrating their suitability for this problem, their effectiveness,

and their computational efficiency. The models are based on judgments regarding the similarity of argument instances with respect to their semantic roles. We showed that these judgments are comparatively simple to formulate and incorporate into a graph representation of the data. We have introduced the idea of separating different similarity features into different graph layers, which resolves the problem faced by many similarity-based approaches of having to heuristically define an instance-wise similarity function and brings the advantage that cluster similarities can be computed in a more principled way. Beyond semantic role labeling, we hope that the multi-layered graph representation described here might be of relevance to other unsupervised problems such part-of-speech tagging or coreference resolution. The approach is general and amenable to other graph partitioning algorithms besides agglomeration and label propagation.

There are two forms of data sparsity that arise in the context of our work, namely, the lexical sparsity of argument head lemmas and the sparsity of specific combinations of linking and syntactic position. As our methods are unsupervised, the conceptually simple solution to sparsity is to train on larger data sets. Because, with some modifications, our graph partitioning approaches could be scaled to larger data sets (in terms of orders of magnitude), this is an obvious next step and would address both instances of data sparsity. Firstly, it would allow us to incorporate a richer set of syntactic features for initialization and would therefore necessarily result in initial clusterings of higher purity. Secondly, the larger size of clusters would result in more reliable similarity scores. Augmenting the data set would therefore almost surely increase the quality of induced clusterings; however, we leave this to future work.

Another interesting future direction would be to eliminate the model's reliance on a syntactic parser that prohibits its application to languages for which parsing resources are not available. It would therefore be worthwhile, albeit challenging, to build models that operate on more readily available forms of syntactic analysis or even raw text. For example, existing work (Abend and Rappoport 2010b; Abend, Reichart, and Rappoport 2009) attempts to identify arguments and distinguish them into core and adjunct ones through unsupervised part of speech and grammar induction. As much as making our model more unsupervised it would also be interesting to see whether some form of *weak supervision* might help induce higher-quality semantic roles without incurring a major labeling effort. The ideas conveyed in this article and the proposed methods extend naturally to this setting: Introducing labels on some of the graph vertices would translate into a semi-supervised graph-based learning task, akin to Zhu, Ghahramani, and Lafferty (2003).

Appendix A. Argument Identification Rules

This appendix specifies the full set of relations used by Rules (2) and (4) of the argument identification rules given for English in Section 5.2, Table 1. The symbols ↑ and ↓ denote the direction of the dependency relation (upward and downward, respectively). The dependency relations are explained in Surdeanu et al. (2008), in their Table 4.

The relations in Rule (2) from Table 1 are IM↑↓, PRT↓, COORD↑↓, P↑↓, OBJ↑, PMOD↑, ADV↑, SUB↑↓, ROOT↑, TMP↑, SBJ↑, OPRD↑.

The relations in Rule (4) are ADV↑↓, AMOD↑↓, APPO↑↓, BNF↑↓-, CONJ↑↓, COORD↑↓, DIR↑↓, DTV↑↓-, EXT↑↓, EXTR↑↓, HMOD↑↓, IOBJ↑↓, LGS↑↓, LOC↑↓, MNR↑↓, NMOD↑↓, OBJ↑↓, OPRD↑↓, POSTHON↑↓, PRD↑↓, PRN↑↓, PRP↑↓, PRT↑↓, PUT↑↓, SBJ↑↓, SUB↑↓, SUFFIX↑↓ TMP↑↓, VOC↑↓ .

Acknowledgments

We are grateful to the anonymous referees, whose feedback helped to substantially improve this article. We also thank the members of the Probabilistic Models reading group at the University of Edinburgh for helpful discussions and comments. We acknowledge the support of EPSRC (grant EP/K017845/1).

References

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

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

Abend, O., R. Reichart, and A. Rappoport. 2009. Unsupervised argument identification for semantic role labeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 28–36, Suntec.

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

Berg-Kirkpatrick, T., A. Bouchard-Côté, J. DeNero, and D. Klein. 2010. Painless unsupervised learning with features. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, CA.

Biemann, C. 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, NY.

Boas, H. 2005. Semantic frames as interlingual representations for multilingual lexical databases. *International Journal of Lexicography*, 18(4):445–478.

Brants, S., S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proceedings of the 1st Workshop on Treebanks and Linguistic Theories*, pages 24–41, Sozopol.

Brigitta, H. 1996. Deutsch ist eine V/2-Sprache mit Verbendstellung und freier Wortfolge. In E. Lang and G. Zifonun, editors, *Deutsch Ű typologisch*, pages 121–141. Walter de Gruyter.

Brown, P. F., V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. 1992. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):283–298.

Burchardt, A., K. Erk, A. Frank, A. Kowalski, S. Padó, and M. Pinkal. 2006. The SALSA corpus: a German corpus resource for lexical semantics. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 969–974, Genoa.

Corston-Oliver, S. and M. Gamon. 2004. Normalizing German and English inflectional morphology to improve statistical word alignment. In Robert Frederking and Kathryn Taylor, editors, *Machine Translation: From Real Users to Research*, volume 3265 of *Lecture Notes in Computer Science*. Springer, Berlin Heidelberg, pages 48–57.

Dowty, D. 1991. Thematic proto roles and argument selection. *Language*, 67(3):547–619.

Fürstenau, H. and M. Lapata. 2009. Graph alignment for semi-supervised semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 11–20, Singapore.

Gamallo, P., A. Agustini, and G. Lopes. 2005. Clustering syntactic positions with similar semantic requirements. *Computational Linguistics*, 31(1):107–146.

Garg, N. and J. Henderson. 2012. Unsupervised semantic role induction with global role ordering. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 145–149, Jeju Island.

Gildea, D. and D. Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Gordon, A. and R. Swanson. 2007. Generalizing semantic role annotations across syntactically similar verbs. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 192–199, Prague.

Gordon, D. and M. Desjardins. 1995. Evaluation and selection of biases in machine learning. *Machine Learning*, 20:5–22.

Grenager, T. and C. Manning. 2006. Unsupervised discovery of a statistical verb lexicon. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, Sydney.

Hajič, J., M. Ciaramita, R. Johansson, D. Kawahara, M. A. Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, P. Straňák, M. Surdeanu, N. Xue, and Y. Zhang. 2009. The CoNLL 2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, CO.

Jain, A., M. Murty, and P. Flynn. 1999. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323.

Kipper, K., H. T. Dang, and M. Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 691–696, Austin, TX.

Koomen, P., V. Punyakanok, D. Roth, and W. Yih. 2005. Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 181–184, Ann Arbor, MI.

Lang, J. and M. Lapata. 2010. Unsupervised induction of semantic roles. In *Proceedings of the North American Chapter of the Association for Computational Linguistics Conference*, pages 939–947, Los Angeles, CA.

Lang, J. and M. Lapata. 2011a. Unsupervised induction of semantic roles via split-merge clustering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1,117–1,126, Portland, OR.

Lang, J. and M. Lapata. 2011b. Unsupervised semantic role induction with graph partitioning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1,320–1,331, Edinburgh.

Levin, B. and M. Rappaport. 2005. *Argument Realization*. Cambridge University Press.

Levin, Beth. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago.

Lin, D. and P. Pantel. 2001. Discovery of inference rules for question-answering. *Natural Langugae Engineering*, 7:343–360.

Manning, C., P. Raghavan, and H. Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Marcus, M., B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Màrquez, L., X. Carras, K. Litkowski, and S. Stevenson. 2008. Semantic role labeling: An introduction to the special issue. *Computational Linguistics*, 34(2):145–159.

Melli, G., 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.

Merlo, P. and S. Stevenson. 2001. Automatic verb classification based on statistical distributions of argument structure. *Computational Linguistics*, 27:373–408.

Munkres, J. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38.

Nivre, J., 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.

Padó, S. and M. Lapata. 2009. Cross-lingual annotation projection of semantic roles. *Journal of Artificial Intelligence Research*, 36:307–340.

Palmer, M., D. Gildea, and P. Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Poon, H. and P. Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Singapore.

Pradhan, S., W. Ward, and J. Martin. 2008. Towards robust semantic role labeling. *Computational Linguistics*, 34(2):289–310.

Ruppenhofer, J., 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.

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

Shen, D. and M. Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21, Prague.

Smith, G. 2003. A Brief Introduction to the TIGER Treebank, Version 1. Technical Report, University of Potsdam.

Surdeanu, M., S. Harabagiu, J. Williams, and P. Aarseth. 2003. Using predicate-argument structures for information extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 8–15, Sapporo.

Surdeanu, M., 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 Conference on Natural Language Learning*, pages 159–177, Manchester.

Swier, R. and S. Stevenson. 2004. Unsupervised semantic role labelling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 95–102, Barcelona.

Titov, I. and A. Klementiev. 2011. A Bayesian model for unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1,445–1,455, Portland, OR.

Titov, I. and A. Klementiev. 2012a. A Bayesian approach to unsupervised semantic role induction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 12–22, Avignon.

Titov, I. and A. Klementiev. 2012b. Crosslingual induction of semantic roles. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 647–656, Jeju Island.

van Rijsbergen, C. 1974. Foundation of evaluation. *Journal of Documentation*, 30(4):265–374.

Wu, D. and P. Fung. 2009. Semantic roles for SMT: A hybrid two-pass model. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 13–16, Boulder, CO.

Zhu, X., Z. Ghahramani, and J. Lafferty. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the International Conference on Machine Learning*, pages 912–919, Washington, DC.