Unsupervised Semantic Role Induction via Split-Merge Clustering

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

Abstract

In this paper we describe an unsupervised method for semantic role induction which holds promise for relieving the data acquisition bottleneck associated with supervised role labelers. We present an algorithm that iteratively splits and merges clusters representing semantic roles, thereby leading from an initial clustering to a final clustering of better quality. The method is simple, surprisingly effective, and allows to integrate linguistic knowledge transparently. By combining role induction with a rule-based component for argument identification we obtain an unsupervised end-to-end semantic role labeling system. Evaluation on the CoNLL 2008 benchmark dataset demonstrates that our method outperforms competitive unsupervised approaches by a wide margin.

1 Introduction

Recent years have seen increased interest in the *shallow semantic analysis* of natural language text. The term is most commonly used to describe the automatic identification and labeling of the semantic roles conveyed by sentential constituents (Gildea and Jurafsky, 2002). Semantic roles describe the relations that hold between a predicate and its arguments, abstracting over surface syntactic configurations. In the example sentences below. *window* occupies different syntactic positions — it is the object of *broke* in sentences (1a,b), and the subject in (1c) — while bearing the same semantic role, i.e., the physical object affected by the breaking event. Analogously, *rock* is the instrument of *break* both when realized as a prepositional phrase in (1a) and as a subject in (1b).

(1) a. $[\text{Joe}]_{\text{A0}}$ **broke** the $[\text{window}]_{\text{A1}}$ with a $[\text{rock}]_{\text{A2}}$.
b. The $[\text{rock}]_{\text{A2}}$ **broke** the $[\text{window}]_{\text{A1}}$.
c. The $[\text{window}]_{\text{A1}}$ **broke**.

The semantic roles in the examples are labeled in the style of PropBank (Palmer et al., 2005), a broad-coverage human-annotated corpus of semantic roles and their syntactic realizations. Under the PropBank annotation framework (which we will assume throughout this paper) each predicate is associated with a set of core roles (named $A0$, $A1$, $A2$, and so on) whose interpretations are specific to that predicate[1] and a set of adjunct roles (e.g., *location* or *time*) whose interpretation is common across predicates. This type of semantic analysis is admittedly shallow but relatively straightforward to automate 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).

Since both argument identification and labeling can be readily modeled as classification tasks, most state-of-the-art systems to date conceptualize se-

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

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1117–1126,
Portland, Oregon, June 19-24, 2011. ©2011 Association for Computational Linguistics

mantic role labeling as a supervised learning problem. Current approaches have high performance — a system will recall around 81% of the arguments correctly and 95% of those will be assigned a correct semantic role (see Màrquez et al. (2008) for details), however only on languages and domains for which large amounts of role-annotated training data are available. For instance, systems trained on PropBank demonstrate a marked decrease in performance (approximately by 10%) when tested on out-of-domain data (Pradhan et al., 2008).

Unfortunately, the reliance on role-annotated data which is expensive and time-consuming to produce for every language and domain, presents a major bottleneck to the widespread application of semantic role labeling. Given the data requirements for supervised systems and the current paucity of such data, unsupervised methods offer a promising alternative. They require no human effort for training thus leading to significant savings in time and resources required for annotating text. And their output can be used in different ways, e.g., as a semantic preprocessing step for applications that require broad coverage understanding or as training material for supervised algorithms.

In this paper we present a simple approach to *unsupervised* semantic role labeling. Following common practice, our system proceeds in two stages. It first identifies the semantic arguments of a predicate and then assigns semantic roles to them. Both stages operate over syntactically analyzed sentences without access to any data annotated with semantic roles. Argument identification is carried out through a small set of linguistically-motivated rules, whereas role induction is treated as a clustering problem. In this setting, 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. We formulate a clustering algorithm that executes a series of *split* and *merge* operations in order to transduce an initial clustering into a final clustering of better quality. Split operations leverage syntactic cues so as to create "pure" clusters that contain arguments of the same role whereas merge operations bring together argument instances of a particular role located in different clusters. We test the effectiveness of our induction method on the CoNLL 2008 benchmark dataset and demonstrate improvements over competitive unsupervised methods by a wide margin.

2 Related Work

As mentioned earlier, much previous work has focused on building supervised SRL systems (Màrquez et al., 2008). A few semi-supervised approaches have been developed within a framework known as *annotation projection*. The idea is to combine labeled and unlabeled data by projecting annotations from a labeled source sentence onto an unlabeled target sentence within the same language (Fürstenau and Lapata, 2009) or across different languages (Padó and Lapata, 2009). Outwith annotation projection, Gordon and Swanson (2007) attempt to increase the coverage of PropBank by leveraging existing labeled data. Rather than annotating new sentences that contain previously unseen verbs, they find syntactically similar verbs and use their annotations as surrogate training data.

Swier and Stevenson (2004) induce role labels with a bootstrapping scheme where the set of labeled instances is iteratively expanded using a classifier trained on previously labeled instances. Their method is unsupervised in that it starts with a dataset containing no role annotations at all. However, it requires significant human effort as it makes use of VerbNet (Kipper et al., 2000) in order to identify the arguments of predicates and make initial role assignments. VerbNet is a broad coverage lexicon organized into verb classes each of which is explicitly associated with argument realization and semantic role specifications.

Abend et al. (2009) propose an algorithm that identifies the arguments of predicates by relying only on part of speech annotations, without, however, assigning semantic roles. In contrast, Lang and Lapata (2010) focus solely on the role induction problem which they formulate as the process of detecting alternations and finding a canonical syntactic form for them. Verbal arguments are then assigned roles, according to their position in this canonical form, since each position references a specific role. Their model extends the logistic classifier with hidden variables and is trained in a manner that makes use of the close relationship between syntactic functions and semantic roles. Grenager and Manning

(2006) propose a directed graphical model which relates a verb, its semantic roles, and their possible syntactic realizations. Latent variables represent the semantic roles of arguments and role induction corresponds to inferring the state of these latent variables.

Our own work also follows the unsupervised learning paradigm. We formulate the induction of semantic roles as a clustering problem and propose a split-merge algorithm which iteratively manipulates clusters representing semantic roles. The motivation behind our approach was to design a conceptually simple system, that allows for the incorporation of linguistic knowledge in a straightforward and transparent manner. For example, arguments occurring in similar syntactic positions are likely to bear the same semantic role and should therefore be grouped together. Analogously, arguments that are lexically similar are likely to represent the same semantic role. We operationalize these notions using a scoring function that quantifies the compatibility between arbitrary cluster pairs. Like Lang and Lapata (2010) and Grenager and Manning (2006) our method operates over syntactically parsed sentences, without, however, making use of any information pertaining to semantic roles (e.g., in form of a lexical resource or manually annotated data). Performing role-semantic analysis without a treebank-trained parser is an interesting research direction, however, we leave this to future work.

3 Learning Setting

We follow the general architecture of supervised semantic role labeling systems. Given a sentence and a designated verb, the SRL task consists of identifying the arguments of the verbal predicate (argument identification) and labeling them with semantic roles (role induction).

In our case neither argument identification nor role induction relies on role-annotated data or other semantic resources although we assume that the input sentences are syntactically analyzed. Our approach is not tied to a specific syntactic representation — both constituent- and dependency-based representations could be used. However, we opted for a dependency-based representation, as it simplifies argument identification considerably and is consistent with the CoNLL 2008 benchmark dataset used for evaluation in our experiments.

Given a dependency 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., $A0$, $A1$).

4 Argument Identification

In the supervised setting, a classifier is employed 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. The rules given in Table 1 are used to discard or select argument candidates. They primarily take into account the parts of speech and the syntactic relations encountered when traversing the dependency tree from predicate to argument. For each candidate, 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 1. Initially, all words save the predicate itself are treated as argument candidates. Then, the rules from Table 1 are applied as follows. Firstly, 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 path ends with an upward-leading OBJ relation (rule (2)). Rule (3) does not match and is therefore not applied. Next, *steady* is discarded because there is a downward-leading OPRD relation along the path and the words *company* and *its* are discarded because of the OBJ relations along the path (rule (4)). Rule (5) does not apply but words *it* and *sales* are kept as likely arguments (rule (6)). Finally, rule (7) does not apply, because there are no candidates left.

1. Discard a candidate if it is a determiner, infinitival marker, coordinating conjunction, or punctuation.
2. Discard a candidate if the path of relations from predicate to candidate ends with coordination, subordination, etc. (see the Appendix 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 the Appendix for the full list of relations).
5. Discard a candidate if it is an auxiliary verb.
6. Keep a candidate if the predicate is its parent.
7. Keep a candidate if the path from predicate to candidate leads along several verbal nodes (verb chain) and ends with arbitrary relation.
8. Discard all remaining candidates.

Table 1: Argument identification rules.

5 Split-Merge Role Induction

We treat role induction as a clustering problem with the goal of assigning argument instances (i.e., specific arguments occurring in an input sentence) to clusters such that these represent semantic roles. In accordance with PropBank, we induce a separate set of clusters for each verb and each cluster thus represents a verb-specific role.

Our algorithm works by iteratively splitting and merging clusters of argument instances in order to arrive at increasingly accurate representations of semantic roles. Although splits and merges could be arbitrarily interleaved, our algorithm executes a single split operation (split phase), followed by a series of merges (merge phase). The split phase partitions the seed cluster containing all argument instances of a particular verb into more fine-grained (sub-)clusters. This initial split results in a clustering with high purity but low collocation, i.e., argument instances in each cluster tend to belong to the same role but argument instances of a particular role are located in many clusters. The degree of dislocation is reduced in the consecutive merge phase, in which clusters that are likely to represent the same role are merged.

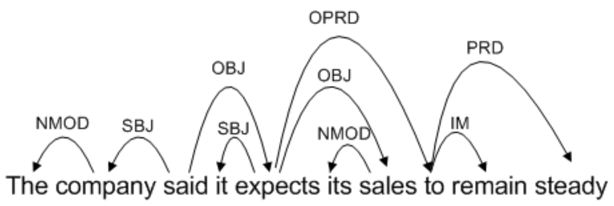

Figure 1: A sample dependency parse with dependency labels SBJ (subject), OBJ (object), NMOD (nominal modifier), OPRD (object predicative complement), PRD (predicative complement), and IM (infinitive marker). See Surdeanu et al. (2008) for more details on this variant of dependency syntax.

5.1 Split Phase

Initially, all arguments of a particular verb are placed in a single cluster. The goal then is to partition this cluster in such a way that the split-off clusters have high purity, i.e., contain argument instances of the same role. Towards this end, we characterize each argument instance by a key, formed by concatenating the following syntactic cues:

- verb voice (active/passive);
- argument linear position relative to predicate (left/right);
- syntactic relation of argument to its governor;
- preposition used for argument realization.

A cluster is allocated for each key and all argument instances with a matching key are assigned to that cluster. Since each cluster encodes fine-grained syntactic distinctions, we assume that arguments occurring in the same position are likely to bear the same semantic role. The assumption is largely supported by our empirical results (see Section 7); the clusters emerging from the initial split phase have a purity of approximately 90%. While the incorporation of additional cues (e.g., indicating the part of speech of the subject or transitivity) would result in even greater purity, it would also create problematically small clusters, thereby negatively affecting the successive merge phase.

5.2 Merge Phase

The split phase creates clusters with high purity, however, argument instances of a particular role are often scattered amongst many clusters resulting in a cluster assignment with low collocation. The goal of the merge phase is to improve collocation by executing a series of merge steps. At each step, pairs of clusters are considered for merging. Each pair is scored by a function that reflects how likely the two clusters are to contain arguments of the same role and the best scoring pair is chosen for merging. In the following, we will specify which pairs of clusters are considered (candidate search), how they are scored, and when the merge phase terminates.

5.2.1 Candidate Search

In principle, we could simply enumerate and score all possible cluster pairs at each iteration. In practice however, such a procedure has a number of drawbacks. Besides being inefficient, it requires a scoring function with comparable scores for arbitrary pairs of clusters. For example, let a, b, c, and d denote clusters. Then, $score(a,b)$ and $score(c,d)$ must be comparable. This is a stronger requirement than demanding that only scores involving some common cluster (e.g., $score(a,b)$ and $score(a,c)$) be comparable. Moreover, it would be desirable to exclude pairings involving small clusters (i.e., with few instances) as scores for these tend to be unreliable. Rather than considering all cluster pairings, we therefore select a specific cluster at each step and score merges between this cluster and certain other clusters. If a sufficiently good merge is found, it is executed, otherwise the clustering does not change. In addition, we prioritize merges between large clusters and avoid merges between small clusters.

Algorithm 1 implements our merging procedure. Each pass through the inner loop (lines 4–12) selects a different cluster to consider at that step. Then, merges between the selected cluster and all larger clusters are considered. The highest-scoring merge is executed, unless all merges are ruled out, i.e., have a score below the threshold α. After each completion of the inner loop, the thresholds contained in the scoring function (discussed below) are adjusted and this is repeated until some termination criterion is met (discussed in Section 5.2.3).

Algorithm 1: Cluster merging procedure. Operation $merge(L_i, L_j)$ merges cluster L_i into cluster L_j and removes L_i from the list L.

```
1  while not done do
2      L ← a list of all clusters sorted by number
       of instances in descending order
3      i ← 1
4      while $i < length(L)$ do
5          $j \leftarrow \arg\max_{0 \le j' < i} score(L_i, L_{j'})$
6          if $score(L_i, L_j) \geq \alpha$ then
7              $merge(L_i, L_j)$
8          end
9          else
10             $i \leftarrow i + 1$
11         end
12     end
13     adjust thresholds
14 end
```

5.2.2 Scoring Function

Our scoring function quantifies whether two clusters are likely to contain arguments of the same role and was designed to reflect the following criteria:

1. whether the arguments found in the two clusters are lexically similar;
2. whether clause-level constraints are satisfied, specifically the constraint that all arguments of a particular clause have different semantic roles, i.e., are assigned to different clusters;
3. whether the arguments present in the two clusters have similar parts of speech.

Qualitatively speaking, criteria (2) and (3) provide negative evidence in the sense that they can be used to rule out incorrect merges but not to identify correct ones. For example, two clusters with drastically different parts of speech are unlikely to represent the same role. However, the converse is not necessarily true as part of speech similarity does not imply role-semantic similarity. Analogously, the fact that clause-level constraints are not met provides evidence against a merge, but the fact that these are satisfied is not reliable evidence in favor of a merge. In contrast, lexical similarity implies that the clusters are likely to represent the same semantic role. It is reasonable to assume that due to selectional restrictions, verbs will be associated with lexical units that are semantically related and assume similar syntactic positions (e.g., *eat* prefers as an object edible things such as *apple, biscuit, meat*), thus bearing the same semantic role. Unavoidably, lexical similarity will be more reliable for arguments with overt lexical content as opposed to pronouns, however this should not impact the scoring of sufficiently large clusters.

Each of the criteria mentioned above is quantified through a separate score and combined into an overall similarity function, which scores two clusters c and c' as follows:

$$score(c,c') = \begin{cases} 0 & \text{if } pos(c,c') < \beta, \\ 0 & \text{if } cons(c,c') < \gamma, \\ lex(c,c') & \text{otherwise.} \end{cases} \quad (2)$$

The particular form of this function is motivated by the distinction between positive and negative evidence. When the part-of-speech similarity (*pos*) is below a certain threshold β or when clause-level constraints (*cons*) are satisfied to a lesser extent than threshold γ, the score takes value zero and the merge is ruled out. If this is not the case, the lexical similarity score (*lex*) determines the magnitude of the overall score. In the remainder of this section we will explain how the individual scores (*pos*, *cons*, and *lex*) are defined and then move on to discuss how the thresholds β and γ are adjusted.

Lexical Similarity We measure lexical similarity between two clusters through cosine similarity. Specifically, each cluster is represented as a vector whose components correspond to the occurrence frequencies of the argument head words in the cluster. The similarity on such vectors x and y is then quantified as:

$$lex(x,y) = cossim(x,y) = \frac{x \cdot y}{\|x\| \|y\|} \quad (3)$$

Clause-Level Constraints Arguments occurring in the same clause cannot bear the same role. Therefore, clusters should not merge if the resulting cluster contains (many) arguments of the same clause. For two clusters c and c' we assess how well they satisfy this clause-level constraint by computing:

$$cons(c,c') = 1 - \frac{2 * viol(c,c')}{NC + NC'} \quad (4)$$

where $viol(c,c')$ refers to the number of pairs of instances $(d,d') \in c \times c'$ for which d and d' occur in the same clause (each instance can participate in at most one pair) and NC and NC' are the number of instances in clusters c and c', respectively.

Part-of-speech Similarity Part-of-speech similarity is also measured through cosine-similarity (equation (3)). Clusters are again represented as vectors x and y whose components correspond to argument part-of-speech tags and values to their occurrence frequency.

5.2.3 Threshold Adaptation and Termination

As mentioned earlier the thresholds β and γ which parametrize the scoring function are adjusted at each iteration. The idea is to start with a very restrictive setting (high values) in which the negative evidence rules out merges more strictly, and then to gradually relax the requirement for a merge by lowering the threshold values. This procedure prioritizes reliable merges over less reliable ones.

More concretely, our threshold adaptation procedure starts with β and γ both set to value 0.95. Then β is lowered by 0.05 at each step, leaving γ unchanged. When β becomes zero, γ is lowered by 0.05 and β is reset to 0.95. Then β is iteratively decreased again until it becomes zero, after which γ is decreased by another 0.05. This is repeated until γ becomes zero, at which point the algorithm terminates. Note that the termination criterion is not tied explicitly to the number of clusters, which is therefore determined automatically.

6 Experimental Setup

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

Data For evaluation purposes, the system's output was compared against the CoNLL 2008 shared task dataset (Surdeanu et al., 2008) which provides

	Syntactic Function			Lang and Lapata			Split-Merge		
	PU	CO	F1	PU	CO	F1	PU	CO	F1
auto/auto	72.9	73.9	73.4	73.2	**76.0**	74.6	**81.9**	71.2	**76.2**
gold/auto	77.7	**80.1**	**78.9**	75.6	79.4	77.4	**84.0**	74.4	**78.9**
auto/gold	77.0	71.0	73.9	77.9	**74.4**	76.2	**86.5**	69.8	**77.3**
gold/gold	81.6	**77.5**	79.5	79.5	76.5	78.0	**88.7**	73.0	**80.1**

Table 2: Clustering results with our split-merge algorithm, the unsupervised model proposed in Lang and Lapata (2010) and a baseline that assigns arguments to clusters based on their syntactic function.

PropBank-style gold standard annotations. The dataset was taken from the Wall Street Journal portion of the Penn Treebank corpus and converted into a dependency format (Surdeanu et al., 2008). In addition to gold standard dependency parses, the dataset also contains automatic parses obtained from the MaltParser (Nivre et al., 2007). Although the dataset provides annotations for verbal and nominal predicate-argument constructions, we only considered the former, following previous work on semantic role labeling (Màrquez et al., 2008).

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

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

$$PU = \frac{1}{N}\sum_i \max_j |G_j \cap C_i| \qquad (5)$$

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

$$CO = \frac{1}{N}\sum_j \max_i |G_j \cap C_i| \qquad (6)$$

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

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

$$F_1 = \frac{2 \times CO \times PU}{CO + PU} \qquad (7)$$

Comparison Models We compared our split-merge algorithm against two competitive approaches. The first one assigns argument instances to clusters according to their syntactic function (e.g., subject, object) as determined by a parser. This baseline has been previously used as point of comparison by other unsupervised semantic role labeling systems (Grenager and Manning, 2006; Lang and Lapata, 2010) and shown difficult to outperform. Our implementation allocates up to $N = 21$ clusters[2] for each verb, one for each of the 20 most frequent functions in the CoNLL dataset and a default cluster for all other functions. The second comparison model is the one proposed in Lang and Lapata (2010) (see Section 2). We used the same model settings (with 10 latent variables) and feature set proposed in that paper. Our method's only parameter is the threshold α which we heuristically set to 0.1. On average our method induces 10 clusters per verb.

7 Results

Our results are summarized in Table 2. We report cluster purity (PU), collocation (CO) and their harmonic mean (F1) for the baseline (Syntactic Function), Lang and Lapata's (2010) model and our split-merge algorithm (Split-Merge) on four

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

		Syntactic Function			Split-Merge		
Verb	Freq	PU	CO	F1	PU	CO	F1
say	15238	91.4	91.3	**91.4**	93.6	81.7	87.2
make	4250	68.6	71.9	70.2	73.3	72.9	**73.1**
go	2109	45.1	56.0	49.9	52.7	51.9	**52.3**
increase	1392	59.7	68.4	63.7	68.8	71.4	**70.1**
know	983	62.4	72.7	**67.1**	63.7	65.9	64.8
tell	911	61.9	76.8	68.6	77.5	70.8	**74.0**
consider	753	63.5	65.6	64.5	79.2	61.6	**69.3**
acquire	704	75.9	79.7	77.7	80.1	76.6	**78.3**
meet	574	76.7	76.0	76.3	88.0	69.7	**77.8**
send	506	69.6	63.8	66.6	83.6	65.8	**73.6**
open	482	63.1	73.4	67.9	77.6	62.2	**69.1**
break	246	53.7	58.9	56.2	68.7	53.3	**60.0**

Table 3: Clustering results for individual verbs with our split-merge algorithm and the syntactic function baseline.

	Syntactic Function			Split-Merge		
Role	PU	CO	F1	PU	CO	F1
A0	74.5	87.0	80.3	79.0	88.7	**83.6**
A1	82.3	72.0	76.8	87.1	73.0	**79.4**
A2	65.0	67.3	66.1	82.8	66.2	**73.6**
A3	48.7	76.7	59.6	79.6	76.3	**77.9**
ADV	37.2	77.3	**50.2**	78.8	37.3	**50.6**
CAU	81.8	74.4	**77.9**	84.8	67.2	75.0
DIR	62.7	67.9	**65.2**	71.0	50.7	59.1
EXT	51.4	87.4	64.7	90.4	87.2	**88.8**
LOC	71.5	74.6	**73.0**	82.6	56.7	67.3
MNR	62.6	58.8	**60.6**	81.5	44.1	57.2
TMP	80.5	74.0	**77.1**	80.1	38.7	52.2
MOD	68.2	44.4	53.8	90.4	89.6	**90.0**
NEG	38.2	98.5	55.0	49.6	98.8	**66.1**
DIS	42.5	87.5	57.2	62.2	75.4	**68.2**

Table 4: Clustering results for individual semantic roles with our split-merge algorithm and the syntactic function baseline.

datasets. These 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). Bold-face is used to highlight the best performing system under each measure on each dataset (e.g., auto/auto, gold/auto and so on).

On all datasets, our method achieves the highest purity and outperforms both comparison models by a wide margin which in turn leads to a considerable increase in F1. On the auto/auto dataset the split-merge algorithm results in 9% higher purity than the baseline and increases F1 by 2.8%. Lang and Lapata's (2010) logistic classifier achieves higher collocation but lags behind our method on the other two measures.

Not unexpectedly, we observe an increase in performance for all models when using gold standard parses. On the gold/auto dataset, F1 increases by 2.7% for the split-merge algorithm, 2.7% for the logistic classifier, and 5.5% for the syntactic function baseline. Split-Merge maintains the highest purity and levels the baseline in terms of F1. Performance also increases if gold standard arguments are used instead of automatically identified arguments. Consequently, each model attains its best scores on the gold/gold dataset.

We also assessed the argument identification component on its own (settings auto/auto and gold/auto). It obtained a precision of 88.1% (percentage of semantic arguments out of those identified) and recall of 87.9% (percentage of identified arguments out of all gold arguments). However, note that these figures are not strictly comparable to those reported for supervised systems, due to the fact that our argument identification component only discards non-argument candidates.

Tables 3 and 4 shows how performance varies across verbs and roles, respectively. We compare the syntactic function baseline and the split-merge system on the auto/auto dataset. Table 3 presents results for 12 verbs which we selected so as to exhibit varied occurrence frequencies and alternation patterns. As can be seen, the macroscopic result — increase in F1 (shown in bold face) and purity — also holds across verbs. Some caution is needed in interpreting the results in Table 4[3] since core roles A0–A3 are defined on a per-verb basis and do not necessarily have a uniform corpus-wide interpretation. Thus, conflating scores across verbs is only meaningful to the extent that these labels actually signify the same

[3]Results are shown for four core roles (A0–A3) and all subtypes of the ArgM role, i.e., adjuncts denoting general purpose (ADV), cause (CAU), direction (DIR), extent (EXT), location (LOC), manner (MNR), and time (TMP), modal verbs (MOD), negative markers (NEG), and discourse connectives (DIS).

role (which is mostly true for A0 and A1). Furthermore, the purity scores given here represent the average purity of those clusters for which the specified role is the majority role. We observe that for most roles shown in Table 4 the split-merge algorithm improves upon the baseline with regard to F1, whereas this is uniformly the case for purity.

What are the practical implications of these results, especially when considering the collocation-purity tradeoff? If we were to annotate the clusters induced by our system, low collocation would result in higher annotation effort while low purity would result in poorer data quality. Our system improves purity substantially over the baselines, without affecting collocation in a way that would massively increase the annotation effort. As an example, consider how our system could support humans in labeling an unannotated corpus. (The following numbers are derived from the CoNLL dataset[4] in the auto/auto setting.) We might decide to annotate all induced clusters with more than 10 instances. This means we would assign labels to 74% of instances in the dataset (excluding those discarded during argument identification) and attain a role classification with 79.4% precision (purity).[5] However, instead of labeling all $165,662$ instances contained in these clusters individually we would only have to assign labels to $2,869$ clusters. Since annotating a cluster takes roughly the same time as annotating a single instance, the annotation effort is reduced by a factor of about 50.

8 Conclusions

In this paper we presented a novel approach to unsupervised role induction which we formulated as a clustering problem. We proposed a split-merge algorithm that iteratively manipulates clusters representing semantic roles whilst trading off cluster purity with collocation. The split phase creates "pure" clusters that contain arguments of the same role whereas the merge phase attempts to increase collocation by merging clusters which are likely to represent the same role. The approach is simple, intuitive and requires no manual effort for training. Coupled with a rule-based component for automatically identifying argument candidates our split-merge algorithm forms an end-to-end system that is capable of inducing role labels without any supervision.

Our approach holds promise for reducing the data acquisition bottleneck for supervised systems. It could be usefully employed in two ways: (a) to create preliminary annotations, thus supporting the "annotate automatically, correct manually" methodology used for example to provide high volume annotation in the Penn Treebank project; and (b) in combination with supervised methods, e.g., by providing useful out-of-domain data for training. An important direction for future work lies in investigating how the approach generalizes across languages as well as reducing our system's reliance on a treebank-trained parser.

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

Appendix

The relations in Rule (2) from Table 1 are IM↑↓, PRT↓, COORD↑↓, P↑↓, OBJ↑, PMOD↑, ADV↑, SUB↑↓, ROOT↑, TMP↑, SBJ↑, OPRD↑. The symbols ↑ and ↓ denote the direction of the dependency arc (upward and downward, respectively).

The relations in Rule (3) 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↑↓. Dependency labels are abbreviated here. A detailed description is given in Surdeanu et al. (2008), in their Table 4.

References

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

[4]Of course, it makes no sense to label this dataset as it is already labeled.

[5]Purity here is slightly lower than the score reported in Table 2 (auto/auto setting), because it is computed over a different number of clusters (only those with at least 10 instances).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

D. Wu and P. Fung. 2009. Semantic Roles for SMT: A Hybrid Two-Pass Model. In *Proceedings of North American Annual Meeting of the Association for Computational Linguistics HLT 2009: Short Papers*, pages 13–16, Boulder, Colorado.