Distributed Representations for Unsupervised Semantic Role Labeling

Kristian Woodsend and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
k.woodsend@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

We present a new approach for unsupervised semantic role labeling that leverages distributed representations. We induce embeddings to represent a predicate, its arguments and their complex interdependence. Argument embeddings are learned from surrounding contexts involving the predicate and neighboring arguments, while predicate embeddings are learned from argument contexts. The induced representations are clustered into roles using a linear programming formulation of hierarchical clustering, where we can model task-specific knowledge. Experiments show improved performance over previous unsupervised semantic role labeling approaches and other distributed word representation models.

1 Introduction

In recent years, an increasing body of work has been devoted to learning distributed word representations and their successful usage in numerous tasks and real-world applications. Examples include language modeling (Collobert et al., 2011; Mikolov et al., 2013c; Mnih and Kavukcuoglu, 2013), paraphrase detection (Socher et al., 2011a), sentiment analysis (Socher et al., 2011b; Kalchbrenner et al., 2014), and most notably machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Auli et al., 2013). Distributed word representations (also known as word embeddings) are trained by predicting the contexts in which the words or phrases occur.

In this paper, we present a new approach for *unsupervised* semantic role labeling that leverages distributed representations. The goal of semantic role labeling is to discover the relations that hold between a predicate and its arguments in a given input sentence (e.g., "who" did "what" to "whom", "when", "where", and "how").

1. [The burglar]$_{\text{A0}}$ [broke]$_{\text{V}}$ [the window]$_{\text{A1}}$.
2. [The window]$_{\text{A1}}$ [broke]$_{\text{V}}$.

In sentence (1), A0 represents the *Agent* of the breaking event, A1 represents the *Patient* (i.e., the physical object affected by the breaking event) and V determines the boundaries of the predicate. The semantic roles in the example are labeled in the style of PropBank (Palmer et al., 2005), a broad-coverage human-annotated corpus of semantic roles and their syntactic realizations. In the unsupervised case, the model must induce such labels from data without access to a predefined set of semantic roles.

Role induction is commonly treated as a clustering problem (Titov and Klementiev, 2012; Lang and Lapata, 2014). The input to the model are instances of arguments (e.g., *window*, *the burglar* in sentence (1)) and the output is a grouping of these instances into clusters such that each cluster contains arguments corresponding to a specific semantic role and each role corresponds to exactly one cluster. In other words, the syntactic representations of verbal predicates, and argument positions are observable, whereas the associated semantic roles are latent and need to be inferred.

The task is challenging due to its unsupervised nature — it is difficult to define a learning objective function whose optimization will yield an accurate model — but also because each predicate can allow several alternate mappings or linkings between its semantic roles and their syntactic realization. Despite occupying different syntactic positions (subject in sentence (1) and object in sentence (2)), the noun phrase *the window* expresses the same role in both sentences. To learn such linkings, previous work has made use of syntactic and semantic features (e.g., whether two arguments are in the same position in the parse tree,

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2482–2491,
Lisbon, Portugal, 17-21 September 2015.

whether they have the same POS-tags, whether they are lexically similar). These features are typically defined on argument instances, without taking the predicate into account, and do not interact but instead are sequentially applied.

In this work we propose to learn these features and their complex interactions (e.g., selectional restrictions) automatically from data. Specifically, we induce embeddings to represent a predicate *and* its arguments. Argument embeddings are learned from surrounding contexts involving the predicate and neighboring arguments. Analogously, predicate embeddings are learned from contexts representing their arguments. Our model learns a rich feature space which can serve as input to any clustering algorithm. We use a linear programming formulation of hierarchical clustering which is advantageous for two reasons. Firstly, expressing clustering as a global optimization problem with an explicit objective function can potentially yield higher quality output compared to greedy algorithms (such as agglomerative clustering). Secondly, through the use of constraints, we can model task-specific knowledge (e.g., semantic roles are unique within a frame). Experimental results show improved performance over both previous unsupervised semantic role labeling approaches and other distributed word representation models.

2 Related Work

Our model is inspired by recent work in learning distributed representations of words (Bengio et al., 2006; Mnih and Hinton, 2008; Collobert et al., 2011; Turian et al., 2010; Mikolov et al., 2013a). In this framework, a neural network is used to predict a word taking into account its context. Words are represented by vectors which are concatenated or averaged in order to form a representation of the context. We induce vector representations to represent each predicate and its argument. As a learning objective, vectors are required to contribute to a prediction task about the target argument in the sentence, given the predicate and a small window of surrounding arguments. Similarly, predicate vectors are learned from the contexts of preceding arguments, and are required to contribute to the prediction of upcoming arguments. Our vectors encode the semantics of arguments, predicates, and their interdependence.

Approaches to unsupervised semantic role labeling follow two main modeling paradigms. Under the the first variant, 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; Lang and Lapata, 2010; Garg and Henderson, 2012). Role induction here corresponds to inferring the state of the latent variables representing the semantic roles of arguments. The second approach is similarity-driven and based on clustering. For instance, Lang and Lapata (2014) induce semantic roles via graph partitioning: each vertex in a graph corresponds to an argument instance of a predicate and edges represent features expressing syntactic or semantic similarity. The graph partitioning problem is solved using task-specific adaptations of label propagation and agglomerative clustering. Titov and Klementiev (2012) propose a Bayesian clustering algorithm based on the Chinese Restaurant Process. Their model encourages similar verbs to have similar linking preferences using a distance-dependent Chinese Restaurant Process prior.

More recently, Titov and Khoddam (2015) propose a reconstruction-error minimization framerwork for unsupervised semantic role induction. Their model consists of two componenets: the encoder (implemented as a log-linear model) predicts roles given syntactic and lexical features, whereas the reconstruction component (implemented as a probabilistic tensor factorization model) recovers argument fillers based on the role predictions, the predicate and other arguments. The two components are estimated jointly to minimize errors in argument reconstruction.

Our work follows the similarity-driven modeling paradigm. Rather than engineering relevant features, we learn them using a neural network and a task-appropriate training objective. We are thus able to model complex interactions between arguments and their predicates without making simplifying assumptions (e.g., that arguments are conditionally independent of each other given the predicate). Our embeddings are largely independent of the clustering algorithm used to induce the semantic roles. We advocate the use of linear programming, which supports the incorporation of linguistic and structural constraints during cluster formation. ILP techniques have been previously applied to several *supervised* NLP tasks, including semantic role labeling (Punyakanok et al., 2008), how-

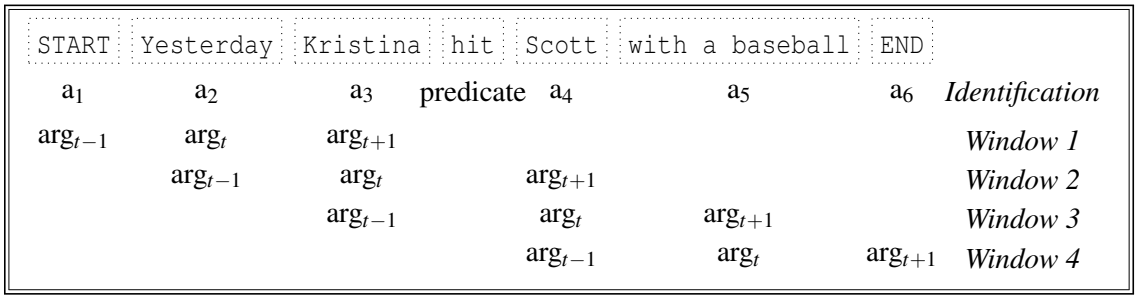

Figure 1: Symmetric context window from the list of arguments

ever their application to *unsupervised* role induction is novel to our knowledge.

3 Model

Unsupervised role induction is commonly modeled after supervised semantic role labeling (Màrquez et al., 2008) and follows a two-stage approach. Given a sentence and a designated verb, the goal is to identify the arguments of the verbal predicate (argument identification) and label them with semantic roles (role induction). The model is first given a syntactically analyzed sentence (e.g., in the form of a dependency parse) with the aim of determining all constitutents that fill a semantic role. Argument identification is performed heuristically using a small number of rules which take into account syntactic relations encountered when traversing the dependency tree from predicate to argument (Lang and Lapata, 2014; Titov and Klementiev, 2012). An alternative which we follow here is to use a supervised classifier trained on a small amount of data using non-lexicalized features.

As mentioned earlier, we treat role induction as a type-level clustering problem: argument instances are assigned to clusters such that these represent semantic roles. We induce a separate set of clusters for each verb, and each cluster thus represents a verb-specific role. Clustering algorithms commonly take a matrix of pairwise similarity scores between instances as input and produce a set of output clusters, often satisfying some optimality criterion. In our case, instances are type-level arguments represented by embeddings whose similarity is quantified using a distance measure such as cosine (see Section 3.1) and clusters are formed using a linear programming formulation of hierarchical clustering (see Section 3.2).

3.1 Predicate and Argument Embeddings

Our approach for learning predicate and argument vectors is inspired by recent methods aimed at learning high-quality vector representations of words from large amounts of unstructured text data (Mikolov et al., 2013a). In this framework, vectors of the surrounding words within a fixed-sized window (the *context*) are summed into a single vector v_c, which is useful in predicting the output vector v_0 representing the current or *target* word. Longer-range context information can also be captured (Le and Mikolov, 2014), specifically words within the current paragraph but outside of the target word context window.

In contrast to previous word-based approaches, our model induces vector representations for each predicate and its semantic arguments. As a learning objective, vectors are required to contribute to a prediction task about the target argument in the sentence, given the predicate and a small window of surrounding arguments. So despite the fact that the argument vectors and weightings are initialized randomly, they can eventually capture semantics as an indirect result of the prediction task. Similarly, predicate vectors are learned from the many contexts sampled from sentences involving that predicate, and are required to contribute to the prediction task of the next argument. One way to consider the role of the predicate token is as another argument. It acts as a memory (similar to the paragraph memory of Le and Mikolov, 2014) that remembers what is missing from the current context, and so captures something of the core nature of the predicate.

Figure 1 illustrates our approach for building the context, for the example sentence *Yesterday, Kristina hit Scott with a baseball.* As a preprocessing step, (verbal) predicates and arguments are identified based on a dependency parse, to give a full list of arguments for the sentence. Boxes show

the span of each argument. In our model, contexts are symmetric and of fixed length ($c = 1$ in the Figure), sampled from a sliding window over the argument list. To enable the first and last arguments within the sentence to be predicted from the context, we augment the argument list with *START* and *END* arguments. Meanwhile, the predicate is associated with all contexts generated from the sliding window approach.

More formally, given a training set comprising a predicate b and a sequence of its semantic arguments $a_1, a_2, a_3, \ldots, a_T$, the objective of the model is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=1}^{T} \log p\left(a_t | b, a_{t+j}, -c \leq j \leq c, j \neq 0\right) \quad (1)$$

where c is the size of the training context around the center argument a_t. We define probability using the softmax function:

$$p\left(a_t | b, a_{\text{context}}\right) \propto \exp\left(v_c^T v_0\right), \quad (2)$$

where v_0 is the target argument vector and v_c the context vector formed from predicate and context arguments vectors. Vectors are trained using stochastic gradient descent where the gradient is obtained via back-propagation. After the training converges, predicates and arguments with similar meaning are mapped to a similar position in the vector space.

Every predicate is mapped to a unique vector v_{pred}, with the vocabulary of vectors shared across the data set. For the arguments, we generate feature vectors f_{-1} and f_{+1} from syntactic information (dependency relations and POS-tags), concatenated with a distributional vector to represent the head word token in each argument. The representation vectors v_{-1} and v_{+1} are calculated from the feature vectors using $v_j = W_{\text{context}} f_j$, where the matrix W_{context} is also updated as part of the learning process. W_{context} is common for all arguments. In a similar manner, the representation vector v_0 for the target argument is calculated using $v_0 = W_{\text{argument}} f_0$. The predicate and argument vectors are concatenated to predict the middle argument in a context. Other ways of dividing the argument window between context and predicted argument, and of combining context vectors, are possible. As the full list of arguments in a sentence is known, we use a symmetric window. The advantage of concatenating the vectors is that information on the sequence of arguments is preserved. An illustration of our model is given in Figure 2.

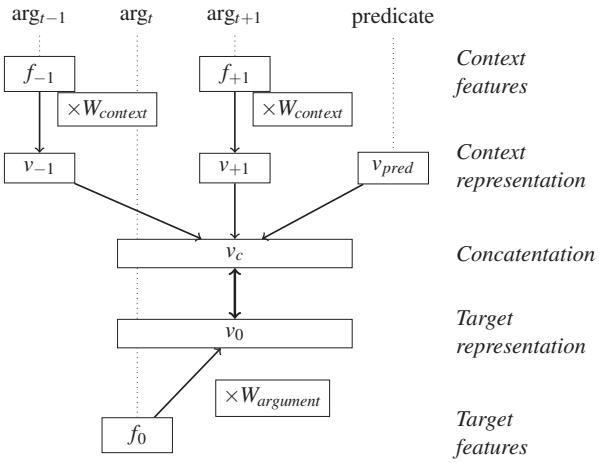

Figure 2: Distributional model for learning representations of predicates and semantic arguments.

Through a context window of arguments rather than neighboring tokens, our model captures a semantic representation of each verbal predicate. Furthermore, the arguments themselves are positioned in vector space as a result of the selectional preferences of the predicates. In the next section, we use the induced semantic space to cluster arguments into semantic roles.

3.2 Argument Clustering

Hierarchical clustering is a method of clustering which seeks to build a hierarchy of clusters, often presented in a dendrogram. In such a representation, all possible pairs of clusters are merged at some level. It is typically implemented as a greedy heuristic algorithm with no explicit objective function. Instead, it requires a measure of dissimilarity between sets of observations, typically through a measure of distance between pairs of observations. An example is the agglomerative clustering technique used in Lang and Lapata (2014). Their algorithm starts from seed clusters based on shared syntactic information, and then repeatedly merges pairs of clusters "bottom up" to form a hierarchy.

It is possible to formalize hierarchical clustering as an integer linear programming (ILP) problem with the dendrogram properties enforced as linear constraints (Gilpin et al., 2013). Although exact solvers exists for ILP, their performance is highly dependent on the number of variables involved, and we found it necessary to develop a linear programming (LP) relaxation to provide approximate solutions faster. Dynamic programming is an al-

ternative approximation technique that could be explored; it has recently been used successfully in the context of supervised semantic role labeling (Täckström et al., 2015).

But first, we consider the exact formalization of agglomerative clustering as an ILP. In order to generate a legal dendrogram, it is necessary for the model to enforce the following partition properties:

Reflexivity A seed cluster is always in the same merged cluster as itself.

Symmetry If seed cluster a is merged into the same cluster as seed cluster b, then b is also in the same cluster as a.

Transitivity If a and b are merged at a certain level, and b and c are also merged at the same level, then a is in the same cluster as c at that level.

To model hierarchical clustering as an ILP problem, we consider all pairs of clusters a and b, and introduce variables $\mathcal{M}_{ab}$ to represent the merge level between clusters a and b. Reflexivity is enforced by the constraint:

$$\mathcal{M}_{aa} = 0, \tag{3}$$

Meanwhile the symmetry requirement is captured by the constraint:

$$\mathcal{M}_{ab} = \mathcal{M}_{ba}. \tag{4}$$

The transitivity requirement and the objective to find the hierarchy that minimizes pair-wise distances are modeled in the objective of the ILP using auxiliary variables O_{abc} that represent the merge order of pairs (a,b) and (a,c), and coefficients w_{abc} that are set equal to the difference between distance metrics D between those pairs:

$$\arg\max_{\mathcal{M},O} \sum_{a,b,c \in \text{Instances}} w_{abc} O_{abc}$$

subject to:

$$\begin{aligned} &\mathcal{M}_{ab} \text{ is a merge function} \\ &O_{abc} = \begin{cases} 1 & \text{if } \mathcal{M}_{ab} < \mathcal{M}_{ac} \\ 0 & \text{otherwise} \end{cases} \\ &w_{abc} = D(a,c) - D(a,b). \end{aligned}$$

Although exact solutions can be found using ILP solvers, for the problems we consider there are typically over 100 seed clusters. This generates in the order of 10^6 transitivity constraints, and it is this in particular that results in combinatorial complexity from off-the-shelf ILP solvers.

An LP relaxation provides approximate solutions faster. A maximum merge level L is first defined as a parameter, although as this is not an integer problem and fractional levels are possible, this does not represent the number of levels. Auxiliary variables $Z_{ab \geq ac}$ capture the merge hierarchy, and T_{abc} rewards transitivity by a factor α:

$$\arg\max_{\mathcal{M},O,Z} \sum_{a,b,c \in \text{Instances}} w_{abc} O_{abc} + \alpha T_{abc}$$

subject to:

$$\begin{aligned} 0 \leq T \leq 1 \\ 0 \leq O \leq 1 \\ 0 \leq Z \leq 1 \\ 0 \leq \mathcal{M} \leq L \\ -L \leq \mathcal{M}_{ac} - \mathcal{M}_{ab} - (L+1) O_{abc} \leq 0 \\ -L \leq \mathcal{M}_{ab} - \mathcal{M}_{ac} - (L+1) Z_{ab \geq ac} + 1 \leq 0 \\ -L \leq \mathcal{M}_{bc} - \mathcal{M}_{ac} - (L+1) Z_{bc \geq ac} + 1 \leq 0 \\ Z_{ab \geq ac} + Z_{bc \geq ac} \geq T_{abc} \end{aligned} \tag{5}$$

To capture the linguistic principles involved in semantic role labeling (Lang and Lapata, 2014), our formulation includes additional constraints. These are expressed explicitly through the construction of the linear programme:

Role Uniqueness Semantic roles are unique within a particular frame. This principle is captured by constraining the merge level of two seed clusters a and b to be at the top level L of the hierarchy, where a and b are roles that occur within the same frame, with the constraint:

$$\mathcal{M}_{ab} = L \qquad \forall (a,b) \text{ in frame.} \tag{6}$$

Syntactic Position Arguments occurring in a specific syntactic position within a specific linking all bear the same semantic role. This is handled by construction of the problem, where all arguments of a particular predicate occurring in a specific syntactic position are collected into a seed cluster at the beginning of the merging problem.

Argument Head Distribution The distribution over argument heads is the same for two clusters that represent the same semantic role. The distribution of arguments is captured in vector space by the model described in Section 3.1. We calculate

centroid vectors from the instances in each cluster. To measure similarity between clusters a and b, we use cosine similarity between centroids:

$$D(a,b) = \frac{v_a^T v_b}{\|v_a\|\|v_b\|} \quad (7)$$

Equations (3)–(7) comprise the LP model.

4 Experimental Setup

In this section we present our experimental setup for assessing the performance of the model presented above. We explain how it was trained and tested, and also briefly introduce the models used for comparison with our approach.

4.1 Training

To obtain distributed representations, we used text from a subset of the English Gigaword corpus (Parker et al., 2011), comprising almost 64 million tokens (2.7 million sentences). The training corpus was pre-processed using `MATE` (Björkelund et al., 2009) to lemmatize the words, provide POS-tags and a dependency parse, identify verbal predicates and the position of arguments.

The neural network model described in Section 3.1 was trained using Matlab. We restricted the predicate vocabulary to use the 5,000 most frequent verbs in the training corpus, and the verbal predicates found in the CoNLL-2008 shared task data set (Surdeanu et al., 2008). Predicates were represented as vectors of size 80, while vectors of length 50 were used for arguments. We used a symmetric context window of size $c = 1$. As the mechanism to prevent all vectors from having the same value, we used "negative-sampling" (Mikolov et al., 2013b), where there are $k = 5$ randomly sampled negative examples of (context, target) pairs for each data sample. This technique has the advantage that we do not need to provide numerical probabilities for the noise distribution. Model parameters were updated during training using stochastic gradient descent over 5 epochs, decreasing the update step size at each epoch.

4.2 Argument clustering

Following common practice in unsupervised role induction (Titov and Klementiev, 2012; Lang and Lapata, 2014), we evaluated our model on the complete CoNLL-2008 shared task data set. We used the clustering metrics of purity, collocation and their harmonic mean F1. In addition, we used V-measure (Rosenberg and Hirschberg, 2007), an entropy-based measure which explicitly evaluates how successfully the criteria of homogeneity and completeness have been satisfied.

In previous work on unsupervised role induction, the results for each predicate were weighted in proportion to the number of times the predicate appeared in the CoNLL-2008 test set. In addition to this measure, we evaluate clustering where predicates are uniformly weighted. In a data set where the top 10 predicates account for almost 20% of the samples, these metrics give a view of performance on the other 3,000-plus predicates where less predicate-specific data is available.

4.3 Comparison Models

We compared our model against a baseline that assigns arguments to clusters based on their syntactic functions (SYNTF; Lang and Lapata, 2014). Specifically, the baseline forms clusters from the syntactic position of an argument using four cues: the verb's voice, the argument's position relative to the predicate, its syntactic relation, and any realizing preposition.[1]

To assess whether our *argument*-based model has any advantages over other *word*-based distributed representations we compared the following variants: (a) the `arg2vec` model presented in Section 3.1 trained on the subset of Gigaword; (b) the continuous bag-of-words model trained using `word2vec` on the same Gigaword corpus; and (c) 300–dimensional vectors pre-trained on part of the Google News dataset[2] (about 100 billion words), again using `word2vec`. In all three instances, we performed argument clustering using the LP of Section 3.2. We also compare against Agglomerative-cosine (AGGLOM), the best performing model of Lang and Lapata (2014).[3] Where applicable, we also refer to the models presented in Titov and Klementiev (2012).

5 Results

Our results on the semantic role induction task are summarized in Tables 1 and 2. Table 1 presents results using the gold standard parses and argu-

[1] Differences in the results compared to Lang and Lapata (2014) are due to our re-implementation of the predicate labeling stage, to be consistent with the preprocessing we used for the other comparison systems.

[2] `http://code.google.com/p/word2vec/`

[3] Differences from published results are again due to changes at the predicate labeling stage.

	Weighted			Unweighted			Weighted			Unweighted		
	PU	CO	F1	PU	CO	F1	HO	CO	V1	HO	CO	V1
SYNTF	81.6	78.1	79.8	90.0	86.8	87.8	71.7	66.2	68.8	85.5	81.7	82.1
AGGLOM	87.4	75.3	**80.9**	95.1	80.7	86.5	79.2	65.5	71.7	93.1	78.1	84.0
word2vec-GIGAWORD	82.8	77.9	80.3	91.4	86.3	88.2	78.8	63.7	70.4	90.2	81.1	84.4
word2vec-GOOGLENEWS	83.4	76.2	79.7	91.6	85.7	87.9	78.7	63.7	70.4	90.2	80.7	84.1
arg2vec-GIGAWORD	87.9	74.7	80.8	94.2	85.4	**88.9**	86.1	64.6	**73.8**	94.6	80.9	**86.2**

Table 1: Purity, collocation and F1 measures (left), and homogeneity, completeness and V1 measures (right) for CoNLL-2008 data set, using gold syntax information.

	Weighted			Unweighted			Weighted			Unweighted		
	PU	CO	F1	PU	CO	F1	HO	CO	V1	HO	CO	V1
SYNTF	68.3	72.1	70.1	80.6	81.3	80.3	55.2	54.9	55.0	74.4	74.3	73.2
AGGLOM	75.5	69.5	72.4	89.3	77.9	82.4	64.9	55.7	60.0	86.1	74.6	78.8
DEPREL+MATE	81.4	77.7	79.5	88.7	84.8	86.2	71.5	65.7	68.5	83.7	79.2	80.3
word2vec-GIGAWORD	83.3	76.1	79.5	91.3	85.7	87.8	78.4	63.4	70.1	89.9	80.3	83.9
word2vec-GOOGLENEWS	82.9	77.4	80.1	91.3	86.0	88.0	78.3	63.6	70.2	89.9	80.9	84.2
arg2vec-GIGAWORD	87.7	74.6	**80.6**	93.9	85.3	**88.8**	85.7	64.4	**73.5**	94.4	80.8	**86.1**

Table 2: Purity, collocation and F1 measures, and homogeneity, completeness and V1 measures for CoNLL-2008 data set using automatic parse syntax information.

ments available in the CoNLL 2008 data set. Notice that our embeddings are still learned using automatically identified arguments. Table 2 uses automatic parses with automatically identified arguments which is a more realistic evaluation setting.

As can be seen in Table 1, when gold standard information is used the syntactic function baseline (SYNTF) is very effective. When considering F1 (weighted by the number of instances), arg2vec performs on the same par with graph-based agglomerative clustering (AGGLOM). Interestingly, word2vec performs worse when trained either on Gigaword or the Google News corpora. According to (weighted) V1, arg2vec outperforms all other comparison models. When predicates are weighted uniformly, arg2vec is the best performing model using F1 or the more information-centric V-measure. This suggests that our model performs well on the the less-frequent predicates and rarer semantic roles. The results also show that our model captures semantic information useful for this task more successfully than the word-based distributional models. Both word2vec models have similar performance, despite significant differences in the size of their training data.

Table 2 shows similar trends. The poorer performance of SYNTF and AGGLOM can partly be ascribed to the heuristics used for argument identification: DEPREL+MATE gives the baseline performance of our dependency parser and argument identification. Nevertheless, when comparing systems that have access to the same preprocessing, our arg2vec model gives the best performance particularly in the information-centric V-measures. Also note, that it seems robust to noise incurred by the automatic parsing and argument identification procedures.

Titov and Klementiev (2012) report a (weighted) F1 of 83.0 on the gold standard CoNLL-2008 dataset, using a coupled model where parameters are shared across verbs and a form of smoothing which replaces argument fillers by lexical cluster ids stemming from Brown et al.'s (1992) algorithm (trained on the RCV1 corpus, about 63 millions words). Our model would presumably benefit from a similar coupling mechanism which we could enforce as a constraint in the ILP. However, we leave this to future work. When tested on automatic parses and gold arguments, their model yields a weighted F1 of 78.8. For comparison, arg2vec obtains an F1 of 80.6 on automatic parses and arguments. Figure 4 shows visualizations of the argument semantic space as captured by the arg2vec-GIGAWORD model, for the predicates *eat* and *win*. Dimensionality reduction was performed by the t-SNE library.[4] The visualization suggests that the model learns similarities beyond simple word contexts.

The evaluation presented so far assesses the quality of the argument representations learned by our model. We also wanted to see whether the predicate embeddings capture meaningful semantic content. Figure 3 shows a visualization

[4] http://lvdmaaten.github.io/tsne/

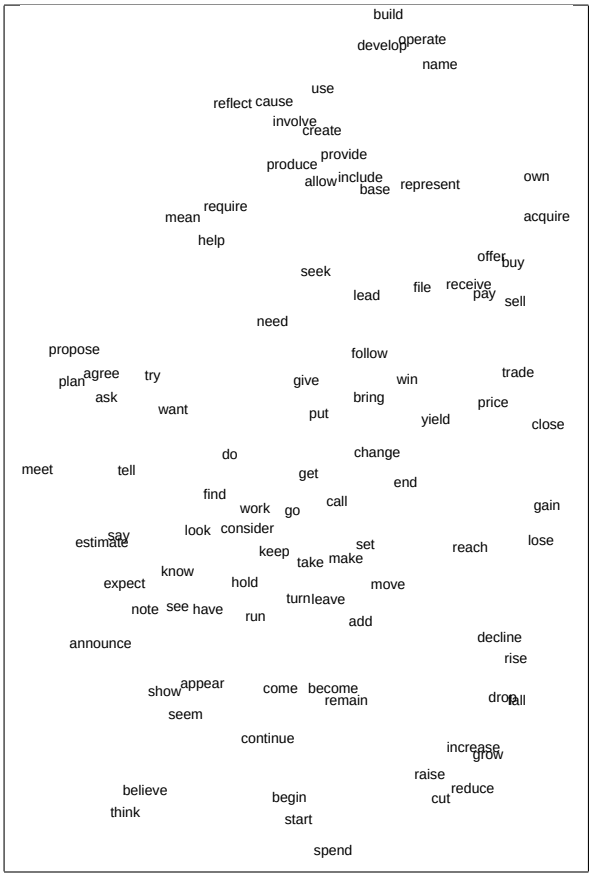

Figure 3: 2-D representation of the induced predicate space for the 100 most frequent predicates in CoNLL-2008.

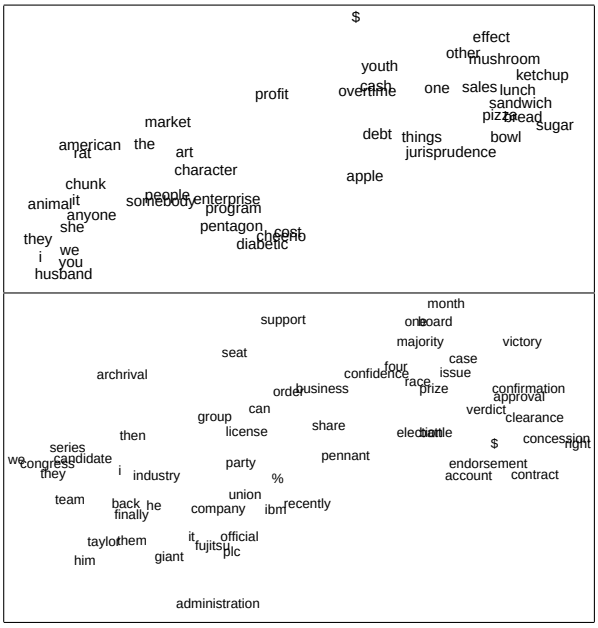

Figure 4: 2-D representation of the induced argument space for the predicates *eat* (top) and *win* (bottom). In both representations, A0 arguments are clustered bottom left, while A1 arguments are found top right.

of the predicate semantic space as captured by `arg2vec` when it is trained on the Gigaword corpus. It shows a projection of the 100 most frequent verbs in CoNLL-2008, with dimensionality reduction again performed by `t-SNE`. The visualization suggests that the model captures non-trivial predicate similarities. Verbs relating to buying and selling lie close together (e.g., *offer, buy, receive, pay, sell*). Verbs denoting growth or decrease are also grouped together (*drop, fall, increase, grow, reduce, cut*). Interestingly, verbs with similar argument structure share regions of the space (e.g., *say, estimate*, or *believe, think* or *seem, appear*). Usefully, verbs are represented in a continuous space rather than discrete clusters (e.g., *acquire* is somewhere between *buy* and *own*).

In order to quantitatively evaluate the quality of the predicate representations induced by our model, we compared the cosine distances between vectors to the hierarchy of VerbNet (Schuler, 2005). VerbNet is a hierarchical domain-independent broad-coverage verb lexicon for English, organizing verbs into classes. The evaluation task was, for all pairs of predicates, to predict whether they would be in the same cluster at the top layer of the hierarchy of VerbNet. To form the top layer of VerbNet, we took the first integer of each VerbNet class number. As an example, the verbs *believe* (VN class conjecture-29.5), *think* (consider-29.9), *expect* (conjecture-29.5-1), and *adopt* (appoint-29.1) would all be in the same class 29. According to this reduction of VerbNet, there are 101 classes. The prediction was based on whether the cosine distance between the pair of vectors was above a threshold value. We measured area under the precision-recall curve (AUC) which captures performance at all thresholds, and F1-score at the best threshold. `arg2vec` does better in both measures than a baseline of random vectors of the same dimension, scoring 0.637 for AUC compared to a baseline of 0.505, and 29.5 against 22.9 for F1.

6 Conclusion

In this paper we presented a new approach for learning distributed representations for predicates and their arguments which we show is useful for unsupervised semantic role labeling. Rather than creating a task-specific algorithm for role induction, we learn a task-specific representation. We

thus decouple feature learning from clustering inference, which results in a conceptually simpler model. Through a formulation of the clustering problem as a linear programme, we are able to perform clustering efficiently and incorporate task-specific constraints. In the future, we would like to investigate how our approach generalizes across languages and tasks.

Acknowledgements We would like to thank Miguel Forte and members of the ILCC at the School of Informatics for their valuable feedback. We acknowledge the support of EPSRC (EP/K017845/1) in the framework of the CHIST-ERA READERS project.

References

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Morin, and Jean-Luc Gauvain, 2006. *Neural Probabilistic Language Models*, pages 137–186. Springer.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual Semantic Role Labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 43–48, Boulder, Colorado, June. Association for Computational Linguistics.

P. F. Brown, 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.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, March.

Nikhil Garg and James 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, Korea.

Sean Gilpin, Siegried Nijssen, and Ian Davidson. 2013. Formalizing Hierarchical Clustering as Integer Linear Programming. In *In Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pages 372–378, Bellevue, Washington.

Trond Grenager and Christopher D. 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, Australia.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland.

Joel Lang and Mirella Lapata. 2010. Unsupervised induction of semantic roles. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 939–947, Los Angeles, California.

Joel Lang and Mirella Lapata. 2014. Similarity-driven semantic role induction via graph partitioning. *Computational Linguistics*, 40(3):133–164.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1188–1196.

Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. 2008. Semantic Role Labeling: An Introduction to the Special Issue. *Computational Linguistics*, 34(2):145–159, June.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient Estimation of Word Representations in Vector Space, January.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeffrey Dean. 2013b. Distributed Representations of Words and Phrases and their Compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., October.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta,

Georgia, June. Association for Computational Linguistics.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, pages 1081–1088.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems 26*, pages 2265–2273.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106, March.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition. *LDC2011T07*. Linguistic Data Consortium.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287, June.

Andrew Rosenberg and Julia Hirschberg. 2007. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Karin Kipper Schuler. 2005. *VerbNet: A broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, University of Pennsylvania.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Andrew Y. Ng, and Christopher D. Manning. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 801–809.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 151–161, Edinburgh, Scotland, UK.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 Shared Task on Joint Parsing of Syntactic and Semantic Dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England, August. Coling 2008 Organizing Committee.

Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Efficient inference and structured learning for semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:29–41.

Ivan Titov and Ehsan Khoddam. 2015. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–10, Denver, Colorado.

Ivan Titov and Alexandre Klementiev. 2012. 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, France.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden.