Dependency-Based Semantic Role Labeling using Convolutional Neural Networks

William R. Foland Jr.
OKRobotGo, Ltd.
5345 Dunraven Circle
Golden, Co, 80403, USA
bill.foland@okrobotgo.com

James H. Martin
Department of Computer Science and
Institute of Cognitive Science
University of Colorado
Boulder, CO 80309
James.Martin@colorado.edu

Abstract

We describe a semantic role labeler with state-of-the-art performance and low computational requirements, which uses convolutional and time-domain neural networks. The system is designed to work with features derived from a dependency parser output. Various system options and architectural details are discussed. Incremental experiments were run to explore the benefits of adding increasingly more complex dependency-based features to the system; results are presented for both in-domain and out-of-domain datasets.

1 Introduction

Semantic role labeling (Gildea and Jurafsky [2002]), the task of identifying and classifying the semantic arguments of verbal and nominal predicates in text, represents one of the most complex NLP tasks to be addressed by supervised machine learning techniques. In the standard supervised approach to building SRL systems, collections of multiway classifiers are trained using annotated corpora such as PropBank (Palmer et al. [2005]). In this approach, classifiers are trained using features derived directly from the original source text, as well as from subsequent syntactic and semantic processing.

As reported in several shared tasks (Carreras and Màrquez [2004],Carreras and Màrquez [2005],Hajič et al. [2009]), SRL systems trained in this manner can achieve high performance. State-of-the-art systems employ classifiers such as support vector machines trained with large numbers of relatively complex combinations of features, often combined with re-ranking based on multiple syntactic analyses. Unfortunately, these approaches have a number of non-trivial limitations including the computational cost of the syntactic parsing and the sparse nature of the complex features on which they rely. This latter limitation is particularly critical since it leads to significant degradation in performance when the trained system is applied to texts from new domains.

However, recent results using multilayer neural networks and pre-trained word embeddings have demonstrated high performance using a much smaller number of minimalist features. The architecture described by Collobert et al. [2011] combines time delay convolutional neural networks (Waibel et al. [1989]) and pre-trained word embeddings for a number of NLP tasks. They develop four components and compare their performance to previous benchmarks, one of which is an SRL system which uses features derived from a phrase-structure parse as input, based on the CoNLL 2005 shared task (Carreras and Màrquez [2005]).

The work described here adopts the basic architecture from Collobert et al. [2011] and explores issues related to the use of this architecture in the context of the CoNLL 2009 shared task. In particular, we present Daisy, a system that (1) employs features derived from dependency parse as input, (2) assigns semantic roles to both verbal and nominal predicates, and (3) automatically assigns word senses to the predicates as described in the CoNLL 2009 shared task (Hajič et al. [2009]).

The following sections will describe the architecture of the Daisy system, present state-of-the-art performance on the CoNLL 2009 shared task, and explore the utility of features derived from dependency parses, including a version of the traditional SRL syntactic path feature.

2 Experimental Setup

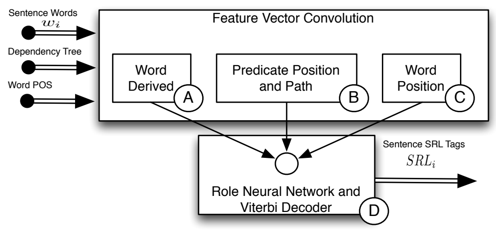
The CoNLL 2009 shared task consists of identifying the sense and semantic arguments for each given argument-bearing token (predicate). In addition to the words themselves, the training data provides the part of speech, syntactic head, and syntactic dependency relation to the head for each word in the sentence. Table 1 shows an example sentence and its representation in the dataset. The PDEPREL and PHEAD features are the head word and dependency relation predicted automatically by a dependency parser. In the example sentence, there are two predicates identified for labeling: *announce*, and *close*. The system should output two arguments for *announce*: *results*:A1 (Object), and *after*:AM-TEMP (Temporal Marker). Similarly, *market*:A1 should be output for the predicate *close*. In addition to role identification, the word sense for each predicate is output, in the example, the expected sense for *announce* is 01, and for *close* is 02.

The training, validation, and evaluation datasets are annotated sentences from the Wall Street Journal. An additional out of domain dataset mostly from the Brown corpus was also supplied. A comprehensive F1 score was generated for both role labels and sense predictions using the provided eval09.pl perl script.

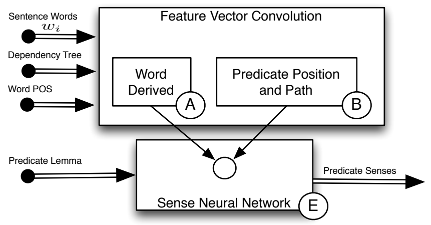
3 Semantic Role Labeling System

The general block diagrams for the Daisy SRL system are shown in Figure 1. The input to the system is a list of words w_i from w_1 to w_n, a list of predicate positions, and dependency parse tree information for the sentence. We treat role labeling and the sense identification as two separate tasks. For each predicate in a given sentence, the Role Subsystem outputs the list of predicted role tags for all words (SRL_i), and the Sense Subsystem outputs the sense tag of the predicate. The system is composed of five major components:

- Word Preprocessing and Word Derived Feature Convolution (Figure 2).
- Predicate Position Feature Convolution.



(a) Role Subsystem



(b) Sense Subsystem

Figure 1: SRL Role and Sense Block Diagrams

- Word Position Feature Convolution.
- Neural Network and Viterbi (Figure 4).
- Predicate Sense Neural Network (Figure 5).

3.1 Word Derived Feature Convolution Section

The Word Derived Features and Convolution section, shown in Figure 2, is sourced by five features which are derived on a word by word basis.

The upper portion of Figure 2 depicts the process of looking up features from the words and parse tree information. The numeric information from the features for each word is concatenated together to form one long feature vector, shown in the diagram as a multi-shaded set of rectangles. Three words of feature information (the word and its two neighbors) from the Word Derived Feature Vector are multiplied by the the weights and bias of Θ_4 and stored in the Convolved Word Derived Feature Vector, for each word in the sentence. For the default convolution width of 300, this results in a long vector of $300 \cdot n$, where n is the number of words in the sentence.

Each feature lookup table contains an entry for *PADDING*. In order to allow the window to extend beyond boundaries of the sentence for early and late words the Feature Vector is padded with the *PADDING* value from each lookup table. If a fea-

ID	FORM	LEMMA	PLEMMA	POS	PPOS	FEAT	PFEAT	HEAD	PHEAD	DEPREL	PDEPREL	FILLPRED	PRED	A[announce]	A[close]
1	The	the	the	DT	DT	_	_	2	2	NMOD	NMOD	_	_	_	_
2	results	result	result	NNS	NNS	_	_	3	3	SBJ	SBJ	_	_	A1	_
3	were	be	be	VBD	VBD	_	_	0	0	ROOT	ROOT	_	_	_	_
4	announced	announce	announce	VBN	VBN	_	_	3	3	VC	VC	Y	announce.01	_	_
5	after	after	after	IN	IN	_	_	4	4	TMP	TMP	_	_	AM-TMP	_
6	the	the	the	DT	DT	_	_	8	8	NMOD	NMOD	_	_	_	_
7	stock	stock	stock	NN	NN	_	_	8	8	NMOD	NMOD	_	_	_	_
8	market	market	market	NN	NN	_	_	9	9	SBJ	SBJ	_	_	_	A1
9	closed	close	close	VBD	VBD	_	_	5	5	SUB	SUB	Y	close.02	_	_
10	.	.	.	.	.	_	_	3	3	P	P	_	_	_	_

Table 1: CoNLL format SRL Dependency Parse Input Test Sentence Example

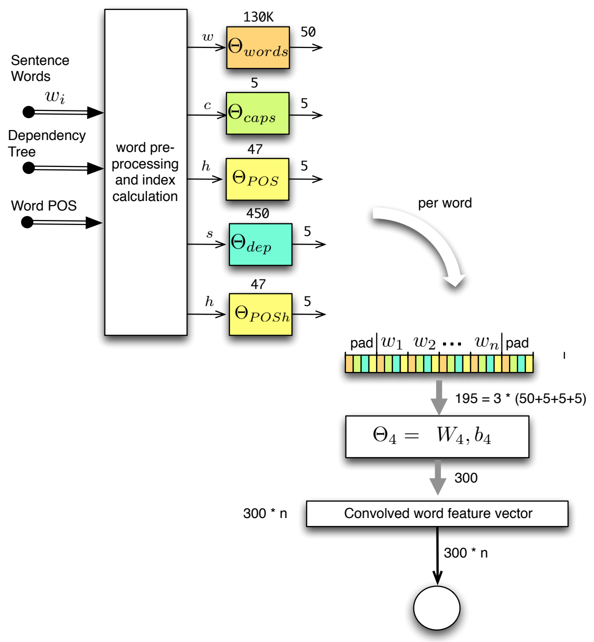


Figure 2: Word Preprocessing, Word Derived Features, and Word Derived Feature Convolution. Ⓐ in figures 1(a) and 1(b).

ture is in the table, the associated vector is output, otherwise the vector corresponding to the special token *UNKNOWN* is output. The *PADDING* and *UNKNOWN* vectors are trained during supervised training.

To train the word representations from scratch, all except the 0.63% least common unique words from the training set are added to the lookup table. The remaining words are therefore trained as the UNKNOWN word, which can then be used to represent any word encountered outside the trained word list. For other features, the representation for the most probable token is used as the UNKNOWN representation.

The five types of word-derived features tested for the SRL Dependency Parse tagger are:

- Word Embeddings
- Capitalization
- POS tag of word
- Dependency Relation
- POS tag of head

3.1.1 Word Pre-processing

The input data provided for the CoNLL 2009 task has already gone through some initial tokenizing. This prevents tokenization differences of different systems from influencing the results, which are meant to allow comparison of the SRL tagging architecture itself. The Daisy pre-processor does not split hyphenated input words, so each input word will result in a single pre-processed word. Numeric values are collapsed to the single common *0* token, and words are lower-cased to create a word representation lookup word.

3.1.2 Word Embeddings

Words are transformed to numeric representations using a lookup table. Like all other feature lookup tables in the system, the word representation vectors can be initialized to small random values to start with, and then trained using the supervised training algorithm.

A method of training the word representations from untagged databases has been very successfully applied to create a starting set of vectors that can be used to initialize a network, which is then fine-tuned with supervised training to execute a specific task. By "pre-training" these word representations using large amounts of untagged text, very informative word relationships can be inexpensively extracted, and later used as the starting point for task specific application learning, see for example Hinton et al. [2006], Bengio et al. [2007] and Weston et al. [2012].

The word representations, or embeddings, used as input to the Daisy SRL System for the experiments described here were generated by Collobert et al. [2011] and were created using a pairwise ranking approach (Schapire and Singer [1998]).

3.1.3 Capitalization

Prior to lower casing, each word is checked for all capitals, initial capital, any capital, or no capitals, and this criteria is used to lookup a vector (default length 5) from the caps table.

3.1.4 Predicted Dependency Relation

The PDEPREL column from the training data, shown in table 1.

3.1.5 Predicted POS tag of word and of head

The Predicted Part-of-speech tag is provided in PPOS column of the training data. The head part of speech tag is found by following the PHEAD column and extracting the PPOS column. (see Table 1).

3.2 Predicate Position and Path Feature Convolution Section

Predicate Position and optional Path features are extracted on a per word basis and convolved, once per predicate (the outer loop of two).

3.2.1 Predicate Position Feature

The position of each word relative to the predicate being evaluated is represented by 25 vectors, based on distances of -12 to +12, and distances outside this range are saturated.

3.2.2 Dependency Path Feature

Information about the path from each word to a given predicate is maintained in a lookup table and is provided in the Predicate Position Convolution section as a per word feature.

Generic Path: The sequence of up and down transitions to traverse the tree from a word to a given predicate is referred to here as the *Generic Path*. The dependency parse trees for each of the two predicates from the example training sentence shown in Table 1 are diagrammed in Figure 3. The Generic Path for each word is shown in the diagram, above the part of speech tag for the word.

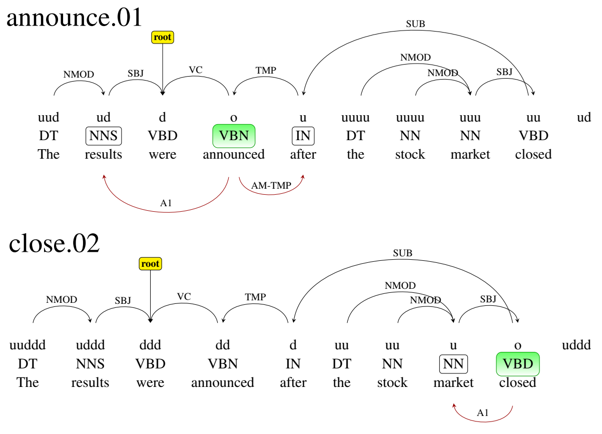

announce.01
root
NMOD
SBJ
VC
TMP
SUB
NMOD
NMOD
SBJ
uud ud d o u uuuu uuuu uuu uu ud
DT NNS VBD VBN IN DT NN NN VBD
The results were announced after the stock market closed
A1
AM-TMP
close.02
root
NMOD
SBJ
VC
TMP
SUB
NMOD
NMOD
SBJ
uuddd uddd ddd dd d uu uu u o uddd
DT NNS VBD VBN IN DT NN NN VBD
The results were announced after the stock market closed
A1

Figure 3: Dependency Parse and Generic Paths

Labeled Path: These are path descriptions which include both the arc direction (Generic Path) and the dependency relation of the arc within the dependency tree. After several rounds of experimentation, we chose to include paths which occur at least five times in the training data, which resulted in about 77K unique path types.

3.3 Word Position Feature Convolution Section

The position of every word with respect to the specific word being evaluated is extracted once per word, per predicate (the inner loop of two). In a similar fashion to the predicate position feature, the position of each word relative to the word being evaluated is represented by 25 vectors, based on distances of -12 to +12, and distances outside this range are saturated.

3.4 Role Neural Network and Viterbi

Figure 4 shows the process of combining the Convolved Feature Vectors, processing with a neural network, and finding the most likely role sequence with a Viterbi detector. Both the Role and Sense neural networks are constructed with a single non-linear layer followed by an output layer. The parameters for each layer are referred to here as Θ, which includes a matrix of weights, W, and a vector of bias terms b. Each layer's output, prior to the activation function, can be calculated from the previous layer's activation output and parameters.

$$f_{\Theta}^{l} = W^{l-1} f_{\Theta}^{l-1} + b^{l-1} \quad (1)$$

The tanh function is used as the nonlinear activation function.

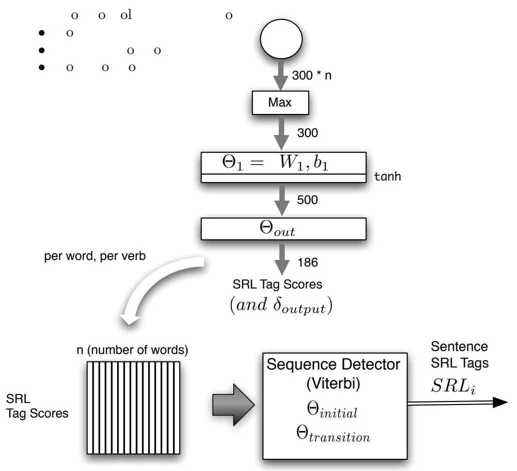


Figure 4: SRL Neural Network and Viterbi. Ⓓ in figure 1(a).

The three Convolved Feature Vectors (diagrammed separately) are summed, then the maximum for each index within each group of 300 is determined. This results in a 300 element vector which will be the input to the Neural Network. A single layer neural network followed by a single output layer is used to create a "score" for each possible role "tag", for the word and predicate being analyzed. After running all words through the system for a single predicate, a matrix of SRL roles scores of size $tags \times words$ is created, which will be used as the input to the Viterbi sequence decoder.

3.5 Sequence Decoder (Viterbi)

The Viterbi decoding algorithm input is a matrix which consists of a vector of SRL role scores for each word. The algorithm is initialized with a learned set of weights per tag, and computes the log-likelihood of transitioning from each state to the next by applying a learned set of weights from the transition matrix.

3.6 Predicate Sense Neural Network

Figure 5 shows the process of combining the Convolved Feature Vectors, processing with a neural network, and finding the most likely sense for a given predicate. The neural network parameters for the sense subsystem are managed with a lookup table holding parameters for each lemma in the training set that is mapped to multiple senses.

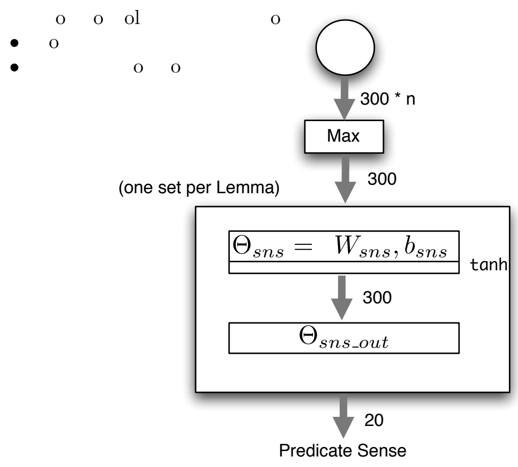


Figure 5: SRL Neural Network for Predicate Sense. Ⓔ in figure1(b).

4 Sense Labeler Training and Forward Model Creation

Both the Role and Sense subsystems are trained using stochastic gradient descent. A forward pass is first run on the system, during which the indices of the maximum values of the sum of the convolutions layers (word-derived and predicate) are saved.

Back-propagation of the Sense Neural Network is based on minimizing a log-likelihood objective:

$$\log p(y|x, \Theta) = f[x, \Theta]_y - \log(\sum_j e^{(f[x,\Theta]_j)}) \quad (2)$$

The two Sense and Role subsystems have the same convolution structures (See figures 1(a) and 1(b)). Experiments run using a common structure for both tasks resulted in about 0.5% worse performance, so the the systems were kept independent.

A separate neural network was trained for each lemma found in the training data set, and the parameters for each network were stored in a lookup table. This results in very large memory requirements during training, especially since Adagrad (Duchi et al.

[2011] was used to decrease training time. To minimize memory requirements and training time, the sense for lemmas which always train to the same sense in the training data are stored in a dictionary. During forward processing, when a lemma is encountered that was not trained (and therefore is not in the parameter lookup table), the sense from the dictionary is output. If the lemma never occurred during training, it won't be in the dictionary, and the most commonly occurring sense of "01" is output by default.

5 Role Labeler Training and Forward Model Creation

During a forward pass, the activation layers and maxIndices are saved and reused during training.

5.1 Cost Calculation

The Viterbi parameters for initial score and transition probabilities are trained using the Sentence Level Log-Likelihood (SLL) cost function.

This cost function is based on Sentence Level Likelihood and is similar to equation 2, except the reference path score must be normalized by using the sum of the exponential of all path scores (the sum of unnormalized probabilities for *all* possible paths, instead of for all possible tags). A recursive method, developed in Rabiner [1989] and specified in Collobert et al. [2011], provides an important and efficient means of computing the sum of the exponential of all path scores. An intermediate vector, δ, is calculated, which will contain the unnormalized log probability that any path through the trellis will pass through a particular state k for the particular word t. The δ vectors have a dimension of N, the number of tags, and they are re-used for the gradient calculation during back-propagation.

5.2 Back-propagation

The recursion described in Collobert et al. [2011] is used to calculate Viterbi delta terms and gradients. The error is then back-propagated through the system in reverse, ending with the feature lookup tables. This is done for each word, for each predicate, requiring two nested loops for training a full sentence. The loop structure makes for long training times, roughly three days on a 2015 compute-optimized AWS core.

6 Results

6.1 Benchmark

The best ConLL 2009, English, SRL F1 score, is labeled Nugues, and the system is described in Björkelund et al. [2009]. To the best of our knowledge, the current state of the art for this dataset is represented by these results, and we therefore use them as a benchmark (See section 7). To generate these benchmark results, 20 features were used for argument identification, including the Dependency Relation Path, and Part of Speech of Dependency Relation Path. A reranker was then run on the output of multiple system outputs.

Table 2 compares the benchmark with a complete Daisy system using a labeled path, with a cutoff of 5, and two separate systems for sense and role labels. F1 scores are 0.41% higher for the WSJ Eval dataset, and 2.59% higher for the out of domain (OOD) Brown dataset.

System Description	WSJ F1	Brown F1
Benchmark (CoNLL2009)	85.63%	73.31%
Daisy	86.04%	75.90%

Table 2: SRL Dependency Parse Test F1

6.2 Metrics

In all experiments, we strictly followed the standard evaluation procedure of the CoNLL 2009 challenge. A simple validation procedure using the specified validation set was used to choose system hyper parameters, and the provided eval09.pl perl script was used to generate all system F1 scores. The system F1 score is the harmonic mean of precision and recall for both role and sense labels. Since we treated the predicate sense disambiguation and the predicate role assignment tasks as independent, it is interesting to view the performance of the two tasks separately. The predicate sense task requires a label for each given predicate, so a per predicate accuracy was calculated (SenseAcc). Similarly we generated a role label F1 score (RoleF1) that is independent of the sense labels. These subsystem performance metrics were also calculated on the CoNLL 2009 benchmark results for comparison.

6.3 Incremental Experiments and Results

Feature abbreviations used in the descriptions are shown in Table 3.

Abbrev.	Feature Description
W	words, initialized randomly prior to training
C	capitalization
P	Part of Speech
HP	Part of Speech of head word
DR	Dependency Relation
GP	Generic path
TW	words, initialized with pre-trained word embeddings prior to training
LP5	Labeled paths that occur at least five times in the training data.

Table 3: Feature Abbreviations

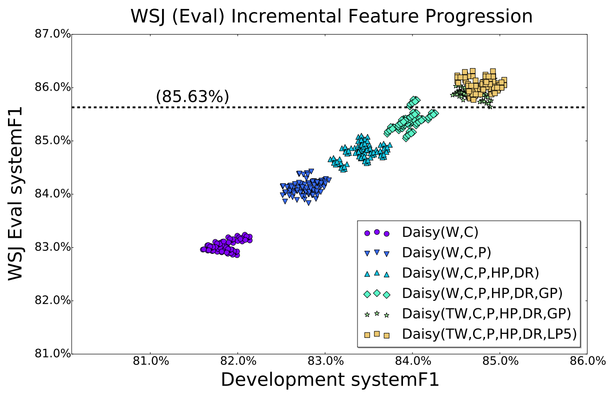


Figure 6: Scatter Plot of Dev F1 vs. Eval F1 for Various Feature Configurations (See also Table 4)

Incremental experiments were run to explore the benefits of adding increasingly more complex dependency-based features to the system.

We began with a basic configuration of only words (randomly initialized) and capitalization (W,C), Following this, a simple per-token part of speech was added (W,C,P). Information from the dependency parser is then added in two steps, first the head word part of speech and dependency relation (W,C,P,HP,DR), and next the generic path (W,C,P,HP,DR,GP). The word representations were then seeded with the pre-trained embeddings described in section 3.1.2 (TW,C,P,HP,DR,GP). Finally, the labeled path was used instead of the generic path, still seeding the words with pre-trained embeddings (TW,C,P,HP,DR,LP5).

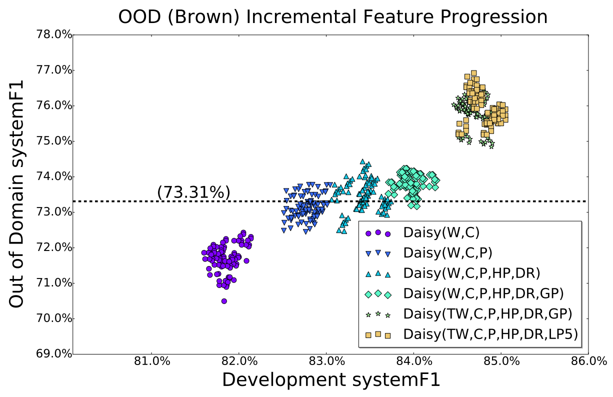


Figure 7: Dev F1 vs. Brown (OOD) F1 for Various Feature Configurations (See also Table 5)

For each system configuration, 12 role subsystems and 8 sense subsystems were trained and tested, using the WSJ development F1 score during training to determine the best model parameter state. After model generation, the WSJ development scores for different systems don't correlate well with the WSJ eval or Brown scores. For example, models with high development scores don't necessarily correspond to best scoring models for the WSJ or Brown data tests.

The CoNLL2009 results used as benchmarks were given as single data points so statistics are not available.

Figure 7 shows the relationship between the development and Evaluation F1 scores, as well as the general performance improvement as features were added.

Tables 4 and 5 show the statistical performance of the system with WSJ and Brown test data.

For the WSJ (evaluation) dataset, the role subsystem F1 improves much more dramatically than the sense subsystem as POS (+1.52%) and dependency parser information (+1.68%) is added. The mean System F1 score is -0.25% under the benchmark without the pre-trained word embeddings. Adding the embeddings boosts performance such that even the lowest scoring systems beat the benchmark, and the mean F1 score is about 0.41% higher.

System Description	SystemF1				RoleF1		SenseAcc	
	Min	Mean	(Δ)	Max	Mean	(Δ)	Mean	(Δ)
Daisy(W,C)	82.86	83.03		83.24	77.47		94.92	
Daisy(W,C,**P**)	83.83	84.12	(+1.09)	84.43	79.00	(+1.52)	95.15	(+0.23)
Daisy(W,C,P,**HP,DR**)	84.46	84.79	(+0.67)	85.10	79.92	(+0.92)	95.29	(+0.13)
Daisy(W,C,P,HP,DR,**GP**)	85.05	85.38	(+0.58)	85.78	80.69	(+0.76)	95.46	(+0.17)
Benchmark (CoNLL2009)		85.63	(+0.25)		81.00	(+0.31)	95.59	(+0.13)
Daisy(**TW**,C,P,HP,DR,GP)	85.64	85.92	(+0.29)	86.17	81.40	(+0.40)	95.66	(+0.07)
Daisy(TW,C,P,HP,DR,**LP5**)	85.77	86.04	(+0.13)	86.31	81.53	(+0.13)	95.77	(+0.11)

Table 4: Performance on WSJ Eval Dataset for Various System Configurations

System Description	SystemF1				RoleF1		SenseAcc	
	Min	Mean	(Δ)	Max	Mean	(Δ)	Mean	(Δ)
Daisy(W,C)	70.50	71.70		72.43	65.49		85.08	
Daisy(W,C,**P**)	72.45	73.13	(+1.43)	73.78	67.38	(+1.89)	85.66	(+0.59)
Benchmark (CoNLL2009)		73.31	(+0.18)		67.78	(+0.40)	85.23	(-0.43)
Daisy(W,C,P,**HP,DR**)	72.47	73.48	(+0.17)	74.43	67.87	(+0.09)	85.71	(+0.48)
Daisy(W,C,P,HP,DR,**GP**)	73.17	73.83	(+0.36)	74.23	68.21	(+0.34)	86.04	(+0.33)
Daisy(**TW**,C,P,HP,DR,GP)	74.85	75.80	(+1.97)	76.46	70.80	(+2.59)	86.72	(+0.68)
Daisy(TW,C,P,HP,DR,**LP5**)	75.19	75.90	(+0.09)	76.93	70.62	(-0.18)	87.40	(+0.69)

Table 5: Performance on Brown Dataset (OOD) for Various System Configurations

For the Brown (OOD) dataset, the role subsystem F1 improves significantly with POS and dependency parse information (+2.72%) while the sense subsystem benefits less (0.96%). The role subsystem dramatically improves when pre-trained words are added (2.59%), due in large part to a better ability to handle unseen words. The mean System F1 scores are higher than the benchmark as soon as dependency parser information is supplied, and the F1 is significantly better for the fully populated system (+2.59%).

7 Related Work

The same Semantic Role Labeling system used to generate the results used as our benchmark was later tested using improved dependency parsing in Björkelund et al. [2010]. Woodsend and Lapata [2014] explore text rewriting and compare results with the benchmark, which they accept as the current state-of-the-art.

Kanerva and Ginter [2014] use the CoNLL 2009 data as a benchmark for investigating the use of Finish and English word vector relationships, and the relationships of word vectors as they relate to semantic roles.

In Socher et al. [2013], the authors present a Recursive Neural Tensor Network (RNTN) which uses word vectors as a primary input and which is used to recursively generate a phrase tree structure for each sentence. The resulting structures are then further used to generate fine-grained sentiment analysis estimates.

Convolutional neural networks which include character level structures have been effectively used for sentiment analysis by dos Santos and Gatti [2014]. The characters are not pre-trained, and syntactic trees are not used as input to the network.

In Luong et al. [2013], words are broken down into morphemes as the input to a recursive neural network to capture morphological compositionality with the goal of improving the vector representations of scarce words.

The characteristics and semantic expressive power of various word embedding collections are investigated by Mikolov et al. [2013] and Chen et al. [2013].

8 Conclusion and Future Work

We have presented a dependency-based semantic role labeler using neural networks, inspired by Collobert et al. [2011] and others to reduce the use of hand-crafted features and make use of unsupervised techniques. Experimental evaluations show that our architecture improves the state of the art performance for this task significantly, for both in domain and out of domain test data. A key element of the system's performance is based on the use of features derived from syntactic dependency parses. The use of a dependency-based path feature, in particular, provides a significant boost in performance over simpler feature sets.

Promising future directions suggested by these results include whether proxies for the dependency-based features can be derived from a similar architecture without the direct need for a full dependency analysis, thus eliminating the pre-processing parser cost. Another future direction involves the predicate disambiguation system. Although this sense disambiguation task is part of the CoNLL 2009 SRL evaluation, it is more properly a word sense disambiguation problem. A more thorough investigation of sense disambiguation in the context of an SRL system is warranted.

References

Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

Anders Björkelund, Love Hafdell, and Pierre Nugues. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–48. Association for Computational Linguistics, 2009.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. A high-performance syntactic and semantic dependency parser. In *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*, pages 33–36. Association for Computational Linguistics, 2010.

Xavier Carreras and Lluís Màrquez. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of the Eighth Conference on Computational Natural Language Learning, CoNLL 2004, Held in cooperation with HLT-NAACL 2004, Boston, Massachusetts, USA, May 6-7, 2004*, pages 89–97, 2004.

Xavier Carreras and Lluís Màrquez. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164. Association for Computational Linguistics, 2005.

Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. The expressive power of word embeddings. *arXiv preprint arXiv:1301.3226*, 2013.

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

Cıcero Nogueira dos Santos and Maıra Gatti. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland*, 2014.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

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

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics, 2009.

Geoffrey Hinton, Simon Osindero, and Yee-Whye

Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

Jenna Kanerva and Filip Ginter. Post-hoc manipulations of vector space models with application to semantic role labeling. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, pages 1–10, 2014.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. Better word representations with recursive neural networks for morphology. *CoNLL-2013*, 104, 2013.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL-HLT*, pages 746–751, 2013.

Martha Palmer, Paul Kingsbury, and Daniel Gildea. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1): 71–106, 2005.

Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

William W Cohen Robert E Schapire and Yoram Singer. Learning to order things. In *Advances in Neural Information Processing Systems 10: Proceedings of the 1997 Conference*, volume 10, page 451. MIT Press, 1998.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

Alex Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(3):328–339, 1989.

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012.

Kristian Woodsend and Mirella Lapata. Text rewriting improves semantic role labeling. *Journal of Artificial Intelligence Research*, pages 133–164, 2014.