Open-Domain Semantic Role Labeling by Modeling Word Spans

Fei Huang
Temple University
1805 N. Broad St.
Wachman Hall 318
fei.huang@temple.edu

Alexander Yates
Temple University
1805 N. Broad St.
Wachman Hall 303A
yates@temple.edu

Abstract

Most supervised language processing systems show a significant drop-off in performance when they are tested on text that comes from a domain significantly different from the domain of the training data. Semantic role labeling techniques are typically trained on newswire text, and in tests their performance on fiction is as much as 19% worse than their performance on newswire text. We investigate techniques for building open-domain semantic role labeling systems that approach the ideal of a train-once, use-anywhere system. We leverage recently-developed techniques for learning representations of text using latent-variable language models, and extend these techniques to ones that provide the kinds of features that are useful for semantic role labeling. In experiments, our novel system reduces error by 16% relative to the previous state of the art on out-of-domain text.

1 Introduction

In recent semantic role labeling (SRL) competitions such as the shared tasks of CoNLL 2005 and CoNLL 2008, supervised SRL systems have been trained on newswire text, and then tested on both an in-domain test set (Wall Street Journal text) and an out-of-domain test set (fiction). All systems tested on these datasets to date have exhibited a significant drop-off in performance on the out-of-domain tests, often performing 15% worse or more on the fiction test sets. Yet the baseline from CoNLL 2005 suggests that the fiction texts are actually easier than the newswire texts. Such observations expose a weakness of current supervised natural language processing (NLP) technology for SRL: systems learn to identify semantic roles for the subset of language contained in the training data, but are not yet good at generalizing to language that has not been seen before.

We aim to build an *open-domain* supervised SRL system; that is, one whose performance on out-of-domain tests approaches the same level of performance as that of state-of-the-art systems on in-domain tests. Importantly, an open-domain system must not use any new labeled data beyond what is included in the original training text when running on a new domain. This allows the system to be ported to any new domain without any manual effort. In particular, it ought to apply to arbitrary Web documents, which are drawn from a huge variety of domains.

Recent theoretical and empirical evidence suggests that the fault for poor performance on out-of-domain tests lies with the representations, or sets of features, traditionally used in supervised NLP. Building on recent efforts in domain adaptation, we develop unsupervised techniques for learning new representations of text. Using latent-variable language models, we learn representations of texts that provide novel kinds of features to our supervised learning algorithms. Similar representations have proven useful in domain-adaptation for part-of-speech tagging and phrase chunking (Huang and Yates, 2009). We demonstrate how to learn representations that are effective for SRL. Experiments on out-of-domain test sets show that our learned representations can dramatically improve out-of-domain performance, and narrow the gap between in-domain and out-of-domain performance by half.

The next section provides background information on learning representations for NLP tasks using latent-variable language models. Section 3 presents our experimental setup for testing open-domain SRL. Sections 4, 5, 6 describe our SRL system: first, how we identify predicates in open-domain text, then how our baseline technique

Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 968–978,
Uppsala, Sweden, 11-16 July 2010.

identifies and classifies arguments, and finally how we learn representations for improving argument identification and classification on out-of-domain text. Section 7 presents previous work, and Section 8 concludes and outlines directions for future work.

2 Open-Domain Representations Using Latent-Variable Language Models

Let $\mathcal{X}$ be an instance set for a learning problem; for SRL, this is the set of all (sentence,predicate) pairs. Let $\mathcal{Y}$ be the space of possible labels for an instance, and let f: $\mathcal{X} \rightarrow \mathcal{Y}$ be the target function to be learned. A *representation* is a function R: $\mathcal{X} \rightarrow \mathcal{Z}$, for some suitable feature space $\mathcal{Z}$ (such as $\mathbb{R}^{\mathrm{d}}$). A *domain* is defined as a distribution $\mathcal{D}$ over the instance set $\mathcal{X}$. An open-domain system observes a set of training examples $(R(x), f(x))$, where instances $x \in \mathcal{X}$ are drawn from a *source* domain, to learn a hypothesis for classifying examples drawn from a separate *target* domain.

Previous work by Ben-David *et al.* (2007; 2009) uses Vapnik-Chervonenkis (VC) theory to prove theoretical bounds on an open-domain learning machine's performance. Their analysis shows that the choice of representation is crucial to open-domain learning. As is customary in VC theory, a good choice of representation must allow a learning machine to achieve low error rates during training. Just as important, however, is that *the representation must simultaneously make the source and target domains look as similar to one another as possible*.

For open-domain SRL, then, the traditional representations are problematic. Typical representations in SRL and NLP use features of the local context to produce a representation. For instance, one dimension of a traditional representation R might be +1 if the instance contains the word "bank" as the head of a noun-phrase chunk that occurs before the predicate in the sentence, and 0 otherwise. Although many previous studies have shown that these features allow learning systems to achieve impressively low error rates during training, they also make texts from different domains look very dissimilar. For instance, a feature based on the word "bank" or "CEO" may be common in a domain of newswire text, but scarce or nonexistent in, say, biomedical literature.

In our recent work (Huang and Yates, 2009) we show how to build systems that learn new representations for open-domain NLP using latent-variable language models like Hidden Markov Models (HMMs). An HMM is a generative probabilistic model that generates each word x_i in the corpus conditioned on a latent variable Y_i. Each Y_i in the model takes on integral values from 1 to K, and each one is generated by the latent variable for the preceding word, Y_{i-1}. The distribution for a corpus $\mathbf{x} = (x_1, \ldots, x_N)$ and a set of state vectors $\mathbf{s} = (s_1, \ldots, s_N)$ is given by:

$$P(\mathbf{x}, \mathbf{s}) = \prod_i P(x_i|s_i)P(s_i|s_{i-1})$$

Using Expectation-Maximization (Dempster et al., 1977), it is possible to estimate the distributions for $P(x_i|s_i)$ and $P(s_i|s_{i-1})$ from unlabeled data. The Viterbi algorithm (Rabiner, 1989) can then be used to produce the optimal sequence of latent states s_i for a given instance $\mathbf{x}$. The output of this process is an integer (ranging from 1 to K) for every word x_i in the corpus. We use the integer value of s_i as a new feature for every x_i in the sentence.

In POS-tagging and chunking experiments, these learned representations have proven to meet both of Ben-David *et al.*'s criteria for open-domain representations: first, they are useful in making predictions on the training text because the HMM latent states categorize tokens according to distributional similarity. And second, it would be difficult to tell two domains apart based on their HMM labels, since the same HMM state can generate similar words from a variety of domains. In what follows, we adapt these representation-learning concepts to open-domain SRL.

3 Experimental Setup

We test our open-domain semantic role labeling system using data from the CoNLL 2005 shared task (Carreras and Màrquez, 2005). We use the standard training set, consisting of sections 02-21 of the Wall Street Journal (WSJ) portion of the Penn Treebank, labeled with PropBank (Palmer et al., 2005) annotations for predicates and arguments. We perform our tests on the Brown corpus (Kucera and Francis, 1967) test data from CoNLL 2005, consisting of 3 sections (ck01-ck03) of propbanked Brown corpus data. This test set consists of 426 sentences containing 7,159 tokens, 804 propositions, and 2,177 arguments. While the

training data contains newswire text, the test sentences are drawn from the domain of "general fiction," and contain an entirely different style (or styles) of English. The data also includes a second test set of in-domain text (section 23 of the Treebank), which we refer to as the WSJ test set and use as a reference point.

Every sentence in the dataset is automatically annotated with a number of NLP pipeline systems, including part-of-speech (POS) tags, phrase chunk labels (Carreras and Màrquez, 2003), named-entity tags, and full parse information by multiple parsers. These pipeline systems are important for generating features for SRL, and one key reason for the poor performance of SRL systems on the Brown corpus is that the pipeline systems themselves perform worse. The Charniak parser, for instance, drops from an F1 of 88.25 on the WSJ test to a F1 of 80.84 on the Brown corpus. For the chunker and POS tagger, the drop-offs are less severe: 94.89 to 91.73, and 97.36 to 94.73.

Toutanova *et al.* (2008) currently have the best-performing SRL system on the Brown corpus test set with an F1 score of 68.81 (80.8 for the WSJ test). They use a discriminative reranking approach to jointly predict the best set of argument boundaries and the best set of argument labels for a predicate. Like the best systems from the CoNLL 2005 shared task (Punyakanok et al., 2008; Pradhan et al., 2005), they also use features from multiple parses to remain robust in the face of parser error. Owing to the established difficulty of the Brown test set and the different domains of the Brown test and WSJ training data, this dataset makes for an excellent testbed for open-domain semantic role labeling.

4 Predicate Identification

In order to perform true open-domain SRL, we must first consider a task which is not formally part of the CoNLL shared task: the task of identifying predicates in a given sentence. While this task is almost trivial in the WSJ test set, where all but two out of over 5000 predicates can be observed in the training data, it is significantly more difficult in an open-domain setting. In the Brown test set, 6.1% of the predicates do not appear in the training data, and 11.8% of the predicates appear at most twice in the training data (*c.f.* 1.5% of the WSJ test predicates that appear at most twice in training). In addition, many words which appear as predicates in training may not be predicates in the test set. In an open-domain setting, therefore, we cannot rely solely on a catalog of predicates from the training data.

	Baseline			HMM		
Freq	P	R	F1	P	R	F1
0	89.1	80.4	84.5	93.5	84.3	**88.7**
0-2	87.4	84.7	86.0	91.6	88.8	**90.2**
all	87.8	92.5	90.1	90.8	96.3	**93.5**

Table 1: Using HMM features in predicate identification reduces error in out-of-domain tests by 34.3% overall, and by 27.1% for OOV predicates. "Freq" refers to frequency in the training data. There were 831 predicates in total; 51 never appeared in training and 98 appeared at most twice.

To address the task of open-domain predicate identification, we construct a Conditional Random Field (CRF) (Lafferty et al., 2001) model with target labels of B-Pred, I-Pred, and O-Pred (for the beginning, interior, and outside of a predicate). We use an open source CRF software package to implement our CRF models.[1] We use words, POS tags, chunk labels, and the predicate label at the preceding and following nodes as features for our Baseline system. To learn an open-domain representation, we then trained an 80 state HMM on the unlabeled texts of the training and Brown test data, and used the Viterbi optimum states of each word as categorical features.

The results of our Baseline and HMM systems appear in Table 1. For predicates that never or rarely appear in training, the HMM features increase F1 by 4.2, and they increase the overall F1 of the system by 3.5 to 93.5, which approaches the F1 of 94.7 that the Baseline system achieves on the in-domain WSJ test set. Based on these results, we were satisfied that our system could find predicates in open-domain text. In all subsequent experiments, we fall back on the standard evaluation in which it is assumed that the boundaries of the predicate are given. This allows us to compare with previous work.

5 Semantic Role Labeling with HMM-based Representations

Following standard practice, we divide the SRL task into two parts: argument identification and

[1] Available from http://sourceforge.net/projects/crf/

argument classification. We treat both sub-tasks as sequence-labeling problems. During argument identification, the system must label each token with labels that indicate either the beginning or interior of an argument (B-Arg or I-Arg), or a label that indicates the token is not part of an argument (O-Arg). During argument classification, the system labels each token that is part of an argument with a class label, such as Arg0 or ArgM. Following argument classification, multi-word arguments may have different classification labels for each token. We post-process the labels by changing them to match the label of the first token. We use CRFs as our models for both tasks (Cohn and Blunsom, 2005).

Most previous approaches to SRL have relied heavily on parsers, and especially constituency parsers. Indeed, when SRL systems use gold standard parses, they tend to perform extremely well (Toutanova et al., 2008). However, as several previous studies have noted (Gildea, 2001; Pradhan et al., 2007), using parsers can cause problems for open-domain SRL. The parsers themselves may not port well to new domains, or the features they generate for SRL may not be stable across domains, and therefore may cause sparse data problems on new domains. Our first step is therefore to build an SRL system that relies on partial parsing, as was done in CoNLL 2004 (Carreras and Màrquez, 2004). We then gradually add in less-sparse alternatives for the syntactic features that previous systems derive from parse trees.

During argument identification we use the features below to predict the label A_i for token w_i:

- *words*: w_i, w_{i-1}, and w_{i+1}
- *parts of speech (POS)*: POS tags t_i, t_{i-1}, and t_{i+1}
- *chunk labels*: (*e.g.*, B-NP, I-VP, or O) chunk tags c_i, c_{i-1}, and c_{i+1}
- *combinations*: $c_i t_i$, $t_i w_i$, $c_i t_i w_i$
- *NE*: the named entity type n_i of w_i
- *position*: whether the word occurs before or after the predicate
- *distance*: the number of intervening tokens between w_i and the target predicate
- *POS before, after predicate*: the POS tag of the tokens immediately preceding and following the predicate
- *Chunk before, after predicate*: the chunk type of the tokens immediately preceding and following the predicate
- *Transition*: for prediction node A_i, we use $A_{i-1}$and A_{i+1} as features

For argument classification, we add the features below to those listed above:

- *arg ID*: the labels A_i produced by arg. identification (B-Arg, I-Arg, or O)
- *combination*: predicate + first argument word, predicate+ last argument word, predicate + first argument POS, predicate + last argument POS
- *head distance*: the number of tokens between the first token of the argument phrase and the target predicate
- *neighbors*: the words immediately before and after the argument.

We refer to the CRF model with these features as our Baseline SRL system; in what follows we extend the Baseline model with more sophisticated features.

5.1 Incorporating HMM-based Representations

As a first step towards an open-domain representation, we use an HMM with 80 latent state values, trained on the unlabeled text of the training and test sets, to produce Viterbi-optimal state values s_i for every token in the corpus. We then add the following features to our CRFs for both argument identification and classification:

- *HMM states*: HMM state values s_i, s_{i-1}, and s_{i+1}
- *HMM states before, after predicate*: the state value of the tokens immediately preceding and following the predicate

We call the resulting model our Baseline+HMM system.

5.2 Path Features

Despite all of the features above, the SRL system has very little information to help it determine the syntactic relationship between a target predicate and a potential argument. For instance, these baseline features provide only crude distance information to distinguish between multiple arguments that follow a predicate, and they make it difficult to correctly identify clause arguments or arguments that appear far from the predicate. Our system needs features that can help distinguish between different syntactic relationships, without being overly sensitive to the domain.

As a step in this direction, we introduce *path* features: features for the sequence of tokens be-

System	P	R	F1
Baseline	63.9	59.7	61.7
Baseline+HMM	68.5	62.7	65.5
Baseline+HMM+Paths	70.0	65.6	67.7
Toutanova *et al.* (2008)	NR	NR	**68.8**

Table 2: Naïve path features improve our baseline, but not enough to match the state-of-the-art. Toutanova *et al.* do not report (NR) separate values for precision and recall on this dataset. Differences in both precision and recall between the baseline and the other systems are statistically significant at $p < 0.01$ using the two-tailed Fisher's exact test.

tween a predicate and a potential argument. In standard SRL systems, these path features usually consist of a sequence of constituent parse nodes representing the shortest path through the parse tree between a word and the predicate (Gildea and Jurafsky, 2002). We substitute paths that do not depend on parse trees. We use four types of paths: word paths, POS paths, chunk paths, and HMM state paths. Given an input sentence labeled with POS tags, and chunks, we construct path features for a token w_i by concatenating words (or tags or chunk labels) between w_i and the predicate. For example, in the sentence "The HIV infection rate is expected to peak in 2010," the word path between "rate" and predicate "peak" would be "is expected to", and the POS path would be "VBZ VBD TO."

Since word, POS, and chunk paths are all subject to data sparsity for arguments that are far from the predicate, we build less-sparse path features by using paths of HMM states. If we use a reasonable number of HMM states, each category label is much more common in the training data than the average word, and paths containing the HMM states should be much less sparse than word paths, and even chunk paths. In our experiments, we use 80-state HMMs.

We call the result of adding path features to our feature set the Baseline+HMM+Paths system((BL). Table 2 shows the performance of our three baseline systems. In this open-domain SRL experiment, path features improve over the Baseline's F1 by 6 points, and by 2.2 points over Baseline+HMM, although the improvement is not enough to match the state-of-the-art system by Toutanova *et al.*

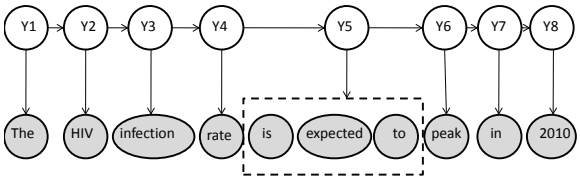

Figure 1: The Span-HMM over the sentence. It shows the span of length 3.

6 Representations for Word Spans

Despite partial success in improving our baseline SRL system with path features, these features still suffer from data sparsity — many paths in the test set are never or very rarely observed during training, so the CRF model has little or no data points from which to estimate accurate parameters for these features. In response, we introduce latent variable models of *word spans*, or sequences of words. As with the HMM models above, the latent states for word spans can be thought of as probabilistic categories for the spans. And like the HMM models, we can turn the word span models into representations by using the state value for a span as a feature in our supervised SRL system. Unlike path features, the features from our models of word spans consist of a single latent state value rather than a concatenation of state values, and as a consequence they tend to be much less sparse in the training data.

6.1 Span-HMM Representations

We build our latent-variable models of word spans using variations of Hidden Markov Models, which we call Span-HMMs. Figure 1 shows a graphical model of a Span-HMM. Each Span-HMM behaves just like a regular HMM, except that it includes one node, called a *span node*, that can generate an entire span rather than a single word. For instance, in the Span-HMM of Figure 1, node y_5 is a span node that generates a span of length 3: "is expected to."

Span-HMMs can be used to provide a single categorical value for any span of a sentence using the usual Viterbi algorithm for HMMs. That is, at test time, we generate a Span-HMM feature for word w_j by constructing a Span-HMM that has a span node for the sequence of words between w_j and the predicate. We determine the Viterbi optimal state of this span node, and use that state as the value of the new feature. In our example in Figure 1, the value of span node y_5 is used as a feature for

the token "rate", since y_5 generates the sequence of words between "rate" and the predicate "peak."

Notice that by using Span-HMMs to provide these features, we have condensed all paths in our data into a small number of categorical values. Whereas there are a huge number of variations to the spans themselves, we can constrain the number of categories for the Span-HMM states to a reasonable number such that each category is likely to appear often in the training data. The value of each Span-HMM state then represents a cluster of spans with similar delimiting words; some clusters will correlate with spans between predicates and arguments, and others with spans that do not connect predicates and arguments. As a result, Span-HMM features are not sparse, and they correlate with the target function, making them useful in learning an SRL model.

6.2 Parameter Estimation

We use a variant of the Baum-Welch algorithm to train our Span-HMMs on unlabeled text. In order for this to work, we need to provide Baum-Welch with a modified view of the data so that span nodes can generate multiple consecutive words in a sentence. First, we take every sentence S in our training data and generate the set $Spans(S)$ of all valid spans in the sentence. For efficiency's sake, we use only spans of length less than 15; approximately 95% of the arguments in our dataset were within 15 words of the predicate, so even with this restriction we are able to supply features for nearly all valid arguments. The second step of our training procedure is to create a separate data point for each span of S. For each span $t \in Spans(S)$, we construct a Span-HMM with a regular node generating each element of S, except that a span node generates all of t. Thus, our training data contains many different copies of each sentence S, with a different Span-HMM generating each copy.

Intuitively, running Baum-Welch over this data means that a span node with state k will be likely to generate two spans t_1 and t_2 if t_1 and t_2 tend to appear in similar contexts. That is, they should appear between words that are also likely to be generated by the same latent state. Thus, certain values of k will tend to appear for spans between predicates and arguments, and others will tend to appear between predicates and non-arguments. This makes the value k informative for both argument identification and argument classification.

6.3 Memory Considerations

Memory usage is a major issue for our Span-HMM models. We represent emission distributions as multinomials over discrete observations. Since there are millions of different spans in our data, a straightforward implementation would require millions of parameters for each latent state of the Span-HMM.

We use two related techniques to get around this problem. In both cases, we use a second HMM model, which we call the base HMM to distinguish from our Span-HMM, to back-off from the explicit word sequence. We use the largest number of states for HMMs that can be fit into memory. Let S be a sentence, and let $\hat{s}$ be the sequence of optimal latent state values for S produced by our base HMM. Our first approach trains the Span-HMM on $Spans(\hat{s})$, rather than $Spans(S)$. If we use a small enough number of latent states in the base HMM (in experiments, we use 10 latent states), we drastically reduce the number of different spans in the data set, and therefore the number of parameters required for our model. We call this representation Span-HMM-Base10. As with our other HMM-based models, we use the largest number of latent states that will allow the resulting model to fit in our machine's memory — our previous experiments on representations for part-of-speech tagging suggest that more latent states are usually better.

While our first technique solves the memory issue, it also loses some of the power of our original Span-HMM model by using a very coarse-grained base HMM clustering of the text into 10 categories. Our second approach trains a separate Span-HMM model for spans of different lengths. Since we need only one model in memory at a time, this allows each one to consume more memory. We therefore use base HMM models with more latent states (up to 20) to annotate our sentences, and then train on the resulting $Spans(\hat{s})$ as before. With this technique, we produce features that are combinations of the state value for span nodes and the length of the span, in order to indicate which of our Span-HMM models the state value came from. We call this representation Span-HMM-BaseByLength.

6.4 Combining Multiple Span-HMMs

So far, our Span-HMM models produce one new feature for every token during argument identifi-

System	P	R	F1
Baseline+HMM+Paths	70.0	65.6	67.7
Toutanova *et al.*	NR	NR	68.8
Span-HMM-Base10	74.5	69.3	71.8
Span-HMM-BaseByLength	76.3	70.2	73.1
Multi-Span-HMM	**77.0**	**70.9**	**73.8**

Table 3: Span-HMM features significantly improve over state-of-the-art results in out-of-domain SRL. Differences in both precision and recall between the baseline and the Span-HMM systems are statistically significant at $p < 0.01$ using the two-tailed Fisher's exact test.

System	WSJ	Brown	Diff
Multi-Span-HMM	79.2	**73.8**	**5.4**
Toutanova *et al.* (2008)	**80.8**	68.8	12.0
Pradhan *et al.* (2005)	78.6	68.4	10.2
Punyakanok *et al.* (2008)	79.4	67.8	11.6

Table 4: Multi-Span-HMM has a much smaller drop-off in F1 than comparable systems on out-of-domain test data vs in-domain test data.

cation and classification. While these new features may be very helpful, ideally we would like our learned representations to produce multiple useful features for the CRF model, so that the CRF can combine the signals from each feature to learn a sophisticated model. Towards this goal, we train N independent versions of our Span-HMM-BaseByLength models, each with a random initialization for the Baum-Welch algorithm. Since Baum-Welch is a hill-climbing algorithm, it should find local, but not necessarily global, optima for the parameters of each Span-HMM-BaseByLength model. When we decode each of the models on training and test texts, we will obtain N different sequences of latent states, one for each locally-optimized model. Thus we obtain N different, independent sources of features. We call the CRF model with these N Span-HMM features the Multi-Span-HMM model(MSH); in experiments we use $N = 5$.

6.5 Results and Discussion

Results for the Span-HMM models on the CoNLL 2005 Brown corpus are shown in Table 3. All three versions of the Span-HMM outperform Toutanova *et al.*'s system on the Brown corpus, with the Multi-Span-HMM gaining 5 points in F1. The Multi-Span-HMM model improves over the Baseline+HMM+Paths model by 7 points in precision, and 5.3 points in recall. Among the Span-HMM models, the use of more states in the Span-HMM-BaseByLength model evidently outweighed the cost of splitting the model into separate versions for different length spans. Using multiple independent copies of the Span-HMMs provides a small (0.7) gain in precision and recall. Differences among the different Span-HMM models were not statistically significant, except that the difference in precision between the Multi-Span-HMM and the Span-HMM-Base10 is significant at $p < .1$.

Table 4 shows the performance drop-off for top SRL systems when applied to WSJ test data and Brown corpus test data. The Multi-Span-HMM model performs near the state-of-the-art on the WSJ test set, and its F1 on out-of-domain data drops only about half as much as comparable systems. Note that several of the techniques used by other systems, such as using features from k-best parses or jointly modeling the dependencies among arguments, are complementary to our techniques, and may boost the performance of our system further.

Table 5 breaks our results down by argument type. Most of our improvement over the Baseline system comes from the core arguments A0 and A1, but also from a few adjunct types like AM-TMP and AM-LOC. Figure 2 shows that when the argument is close to the predicate, both systems perform well, but as the distance from the predicate grows, our Multi-Span-HMM system is better able to identify and classify arguments than the Baseline+HMM+Paths system.

Table 6 provides results for argument identification and classification separately. As Pradhan *et al.*previously showed (Pradhan et al., 2007), SRL systems tend to have an easier time with porting argument identification to new domains, but are less strong at argument classification on new domains. Our baseline system decreases in F-score from 81.5 to 78.9 for argument identification, but suffers a much larger 8% drop in argument classification. The Multi-Span-HMM model improves over the Baseline in both tasks and on both test sets, but the largest improvement (6%) is in argument classification on the Brown test set.

To help explain the success of the Span-HMM techniques, we measured the sparsity of our path

	Overall	A0	A1	A2	A3	A4	ADV	DIR	DIS	LOC	MNR	MOD	NEG	PNC	TMP	R-A0	R-A1
Num	2177	566	676	147	12	15	143	53	22	85	110	91	50	17	112	25	21
BL	67.7	76.2	70.6	64.8	59.0	71.2	52.7	54.8	71.9	67.5	58.3	90.9	90.0	50.0	76.5	76.5	71.3
MSH	73.8	82.5	73.6	63.9	60.3	73.3	50.8	52.9	70.0	70.3	52.7	94.2	92.9	51.6	81.6	84.4	75.7

Table 5: SRL results (F1) on the Brown test corpus broken down by role type. BL is the Baseline+HMM+Paths model, MSH is the Multi-Span-HMM model. Column 8 to 16 are all adjuncts (AM-). We omit roles with ten or fewer examples.

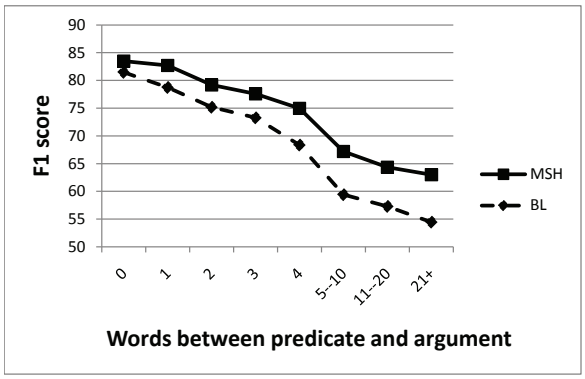

Figure 2: The Multi-Span-HMM (MSH) model is better able to identify and classify arguments that are far from the predicate than the Baseline+HMM+Paths (BL) model.

	Test	Id.F1	Accuracy
BL	WSJ	81.5	93.7
	Brown	78.9	85.8
MSH	WSJ	83.9	94.4
	Brown	80.3	91.9

Table 6: Baseline (BL) and Multi-Span-HMM (MSH) performance on argument identification (Id.F1) and argument classification.

and Span-HMM features. Figure 3 shows the percentage of feature values in the Brown corpus that appear more than twice, exactly twice, or exactly once in the training data. While word path features can be highly valuable when there is training data available for them, only about 11% of the word paths in the Brown test set also appeared at all in the training data. POS and chunk paths fared a bit better (22% and 33% respectively), but even then nearly 70% of all feature values had no available training data. HMM and Span-HMM-Base10 paths achieved far better success in this respect. Importantly, the improvement is mostly due to features that are seen *often* in training, rather than features that were seen just once or twice. Thus Span-

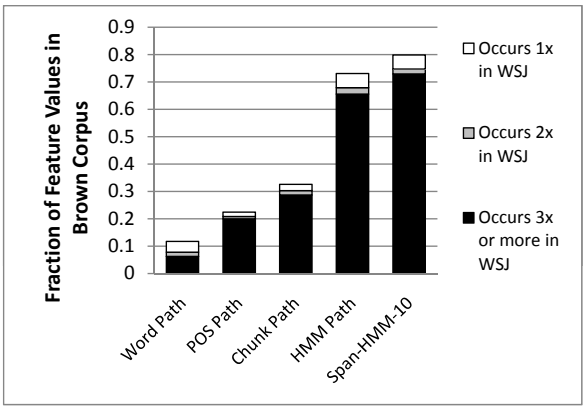

Figure 3: HMM path and Span-HMM features are far more likely to appear often in training data than the word, POS, and chunk path features. Over 70% of Span-HMM-Base10 features in the Brown corpus appear at least three times during training; in contrast, fewer than 33% of chunk path features in the Brown corpus appear at all during training.

HMMs derive their power as representations for open-domain SRL from the fact that they provide features that are mostly the same across domains; 80% of the features of our Span-HMM-Base10 in the Brown corpus were observed at least once in the training data.

Table 7 shows examples of spans that were clustered into the same Span-HMM state, along with word to either side. All four examples are cases where the Span-HMM-Base10 model correctly tagged the following argument, but the Baseline+HMM+Paths model did not. We can see that the paths of these four examples are completely different, but the words surrounding them are very similar. The emission from a span node are very sparse, so the Span-HMM has unsurprisingly learned to cluster spans according to the HMM states that precede and follow the span node. This is by design, as this kind of distributional clustering is helpful for identifying and classifying arguments. One potentially interesting

Predicate	Span	B-Arg
picked	the things up	from
passed	through the barbed wire	at
come	down from Sundays	to
sat	over his second rock	in

Table 7: Example spans labeled with the same Span-HMM state. The examples are taken from sentences where the Span-HMM-Base10 model correctly identified the argument on the right, but the Baseline+HMM+Paths model did not.

question for future work is whether a less sparse model of the spans themselves, such as a Naïve Bayes model for the span node, would yield a better clustering for producing features for semantic role labeling.

7 Previous Work

Deschact and Moens (2009) use a latent-variable language model to provide features for an SRL system, and they show on CoNLL 2008 data that they can significantly improve performance when little labeled training data is available. They do not report on out-of-domain tests. They use HMM language models trained on unlabeled text, much like we use in our baseline systems, but they do not consider models of word spans, which we found to be most beneficial. Downey *et al.* (2007b) also incorporate HMM-based representations into a system for the related task of Web information extraction, and are able to show that the system improves performance on rare terms.

Fürstenau and Lapata (2009b; 2009a) use semi-supervised techniques to automatically annotate data for previously unseen predicates with semantic role information. This task differs from ours in that it focuses on previously unseen predicates, which may or may not be part of text from a new domain. Their techniques also result in relatively lower performance (F1 between 15 and 25), although their tests are on a more difficult and very different corpus. Weston *et al.* (2008) use deep learning techniques based on semi-supervised embeddings to improve an SRL system, though their tests are on in-domain data. Unsupervised SRL systems (Swier and Stevenson, 2004; Grenager and Manning, 2006; Abend et al., 2009) can naturally be ported to new domains with little trouble, but their accuracy thus far falls short of state-of-the-art supervised and semi-supervised systems.

The disparity in performance between in-domain and out-of-domain tests is by no means restricted to SRL. Past research in a variety of NLP tasks has shown that parsers (Gildea, 2001), chunkers (Huang and Yates, 2009), part-of-speech taggers (Blitzer et al., 2006), named-entity taggers (Downey et al., 2007a), and word sense disambiguation systems (Escudero et al., 2000) all suffer from a similar drop-off in performance on out-of-domain tests. Numerous domain adaptation techniques have been developed to address this problem, including self-training (McClosky et al., 2006) and instance weighting (Bacchiani et al., 2006) for parser adaptation and structural correspondence learning for POS tagging (Blitzer et al., 2006). Of these techniques, structural correspondence learning is closest to our technique in that it is a form of representation learning, but it does not learn features for word spans. None of these techniques have been successfully applied to SRL.

8 Conclusion and Future Work

We have presented novel representation-learning techniques for building an open-domain SRL system. By incorporating learned features from HMMs and Span-HMMs trained on unlabeled text, our SRL system is able to correctly identify predicates in out-of-domain text with an F1 of 93.5, and it can identify and classify arguments to predicates with an F1 of 73.8, outperforming comparable state-of-the-art systems. Our successes so far on out-of-domain tests bring hope that supervised NLP systems may eventually achieve the ideal where they no longer need new manually-labeled training data for every new domain. There are several potential avenues for further progress towards this goal, including the development of more portable SRL pipeline systems, and especially parsers. Developing techniques that can incrementally adapt to new domains without the computational expense of retraining the CRF model every time would help make open-domain SRL more practical.

Acknowledgments

We wish to thank the anonymous reviewers for their helpful comments and suggestions.

References

Omri Abend, Roi Reichart, and Ari Rappoport. 2009. Unsupervised argument identification for semantic role labeling. In *Proceedings of the ACL*.

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2007. Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems 20*, Cambridge, MA. MIT Press.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jenn Wortman. 2009. A theory of learning from different domains. *Machine Learning*, (to appear).

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

Xavier Carreras and Lluís Màrquez. 2003. Phrase recognition by filtering and ranking with perceptrons. In *Proceedings of RANLP-2003*.

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*.

Trevor Cohn and Phil Blunsom. 2005. Semantic role labelling with tree conditional random fields. In *Proceedings of CoNLL*.

Arthur Dempster, Nan Laird, and Donald Rubin. 1977. Likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Koen Deschacht and Marie-Francine Moens. 2009. Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

D. Downey, M. Broadhead, and O. Etzioni. 2007a. Locating complex named entities in web text. In *Procs. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*.

Doug Downey, Stefan Schoenmackers, and Oren Etzioni. 2007b. Sparse information extraction: Unsupervised language models to the rescue. In *ACL*.

G. Escudero, L. Márquez, and G. Rigau. 2000. An empirical study of the domain dependence of supervised word sense disambiguation systems. In *EMNLP/VLC*.

Hagen Fürstenau and Mirella Lapata. 2009a. Graph alignment for semi-supervised semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 11–20.

Hagen Fürstenau and Mirella Lapata. 2009b. Semi-supervised semantic role labeling. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 220–228.

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

Daniel Gildea. 2001. Corpus Variation and Parser Performance. In *Conference on Empirical Methods in Natural Language Processing*.

Trond Grenager and Christopher D Manning. 2006. Unsupervised discovery of a statistical verb lexicon. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Fei Huang and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence labeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

H. Kucera and W.N. Francis. 1967. *Computational Analysis of Present-Day American English*. Brown University Press.

J. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 337–344.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: A corpus annotated with semantic roles. *Computational Linguistics Journal*, 31(1).

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*.

Sameer Pradhan, Wayne Ward, and James H. Martin. 2007. Towards robust semantic role labeling. In *Proceedings of NAACL-HLT*, pages 556–563.

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.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.

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

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Jason Weston, Frederic Ratle, and Ronan Collobert. 2008. Deep learning via semi-supervised embedding. In *Proceedings of the 25th International Conference on Machine Learning*.