Adapting Text instead of the Model: An Open Domain Approach

Gourab Kundu, Dan Roth
University of Illinois at Urbana Champaign
Urbana, IL 61801
{kundu2,danr}@illinois.edu

Abstract

Natural language systems trained on labeled data from one domain do not perform well on other domains. Most adaptation algorithms proposed in the literature train a new model for the new domain using unlabeled data. However, it is time consuming to retrain big models or pipeline systems. Moreover, the domain of a new target sentence may not be known, and one may not have significant amount of unlabeled data for every new domain.

To pursue the goal of an *Open Domain NLP* (train once, test anywhere), we propose *ADUT (ADaptation Using label-preserving Transformation)*, an approach that avoids the need for retraining and does not require knowledge of the new domain, or any data from it. Our approach applies simple label-preserving transformations to the target text so that the transformed text is more similar to the training domain; it then applies the existing model on the transformed sentences and combines the predictions to produce the desired prediction on the target text. We instantiate *ADUT* for the case of Semantic Role Labeling (SRL) and show that it compares favorably with approaches that retrain their model on the target domain. Specifically, this "on the fly" adaptation approach yields 13% error reduction for a single parse system when adapting from the news wire text to fiction.

1 Introduction

In several NLP tasks, systems trained on annotated data from one domain perform well when tested on the same domain but adapt poorly to other domains. For example, all systems of CoNLL 2005 shared task (Carreras and Màrquez, 2005) on Semantic Role Labeling showed a performance degradation of almost 10% or more when tested on a different domain.

Most works in domain adaptation have focused on learning a common representation across training and test domains (Blitzer et al., 2006; DauméIII, 2007; Huang and Yates, 2009). Using this representation, they retrain the model for every new domain. But these are not *Open Domain Systems* since the model needs to be retrained for every new domain. This is very difficult for pipeline systems like SRL where syntactic parser, shallow parser, POS tagger and then SRL need to be retrained. Moreover, these methods need to have a lot of unlabeled data that is taken from the same domain, in order to learn meaningful feature correspondences across training and test domain. These approaches cannot work when they do not have a lot of unlabeled data from the test domain or when the test domain in itself is very diverse, e.g., the web.

The contribution of this paper is a new framework for adaptation. We propose *ADUT (ADaptation Using label-preserving Transformation)* as a framework in which a previously learned model can be used on an out-of-domain example without retraining and without looking at any labeled or unlabeled data for the domain of the new example. The framework transforms the test sentence to generate sentences that have, in principle, identical labeling but that are more like instances from the training domain. Consequently, it is expected that the exist-

Proceedings of the Fifteenth Conference on Computational Natural Language Learning, pages 229–237,
Portland, Oregon, USA, 23–24 June 2011.

ing model will make better predictions on them. All these predictions are then combined to choose the most probable and consistent prediction for the test sentence.

ADUT is a general technique which can be applied to any natural language task. In this paper, we demonstrate its usefulness on the task of semantic role labeling (Carreras and Màrquez, 2005). Starting with a system that was trained on the news text and does not perform well on fiction, we show that ADUT provides significant improvement on fiction, and is competitive with the performance of algorithms that were re-trained on the test domain.

The paper is organized as follows. Section 2 discusses two motivating examples. Section 3 gives a formal definition of our adaptation framework. Section 4 describes the transformation operators that we applied for this task. Section 5 presents our joint inference approach. Section 6 describes our semantic role labeling system and our experimental results are in Section 7. Section 8 describes the related works for domain adaptation. Finally in Section 9 we conclude the paper with a discussion.

2 Motivating Examples

One of the key reasons for performance degradation of an NLP tool is unseen features such as words in the new domain that were not seen in the training domain. But if an unknown word is replaced by a known word without changing the labeling of the sentence, tools perform better. For example, in the task of syntactic parsing, the unknown word *checkup* causes the Charniak parser to make a wrong coordination decision on the sentence

> *He was discharged from the hospital after a two-day checkup and he and his parents had what Mr. Mckinley described as a "celebration lunch" at the cafeteria on the campus.*

If we replace the word *checkup* with its hypernym *examination* which appears in training data, the parse gets corrected. Figure 1 shows both original and corrected parse trees.

For the task of semantic role labeling, systems do not perform well on the predicates that are infrequent in training domain. But if an infrequent predicate is replaced with a frequent predicate from training domain such that both predicates have similar semantic argument structure, the system performs better. Consider the following sentence

> *Scotty gazed out at ugly gray slums.*

The semantic role for the phrase *at ugly gray slums* with respect to predicate *gaze* is *A1*. But the predicate *gaze* appears only once in training data and our model predicts *at ugly gray slums* as *AM-LOC* instead of *A1*. But if *gaze* is replaced with *look* which occurs 328 times in training data and has similar argument structure (in the same VerbNet class as *gaze*), the system makes the correct prediction.

3 Problem Formulation

Let the in-domain distribution be D_i and out-of-domain distribution be D_o. We have a model f trained over a set of labeled examples drawn from D_i. If D_i and D_o are very dissimilar, f will not perform well on examples drawn from D_o. The problem is to get good performance from f on D_o without retraining f.

We define a *Transformation* g to be a function that maps an example e into a set of examples E. So $g : X \rightarrow 2^X$ where X is the entire space of examples. In this paper, we only consider the *Label-preserving Transformations* which satisfy the property that all transformed examples in E have the same label as input example e, i.e., $\forall x\ x \in S_k \Rightarrow g(x) \subset S_k$ where S_k is the set of examples with label k . Let G be a set of label-preserving transformation functions. $G = \{g_1, g_2, \ldots, g_p\}$.

At evaluation time, for test example d, we will apply G to get a set of examples T_1. Let $T_2 = \{d' \in T_1 : D_i(d') > D_i(d)\}$. So all examples in T_2 have same label as d but have a higher probability than d to be drawn from the in-domain distribution. So f should perform better on examples in T_2 than on d. For each $d' \in T_2$, f will produce scores for the output labels. The scores will be combined subject to constraints to produce the final output.

4 Transformation Functions

After applying a transformation function to get a new sentence from an input sentence, we remember the mapping of segments across the original

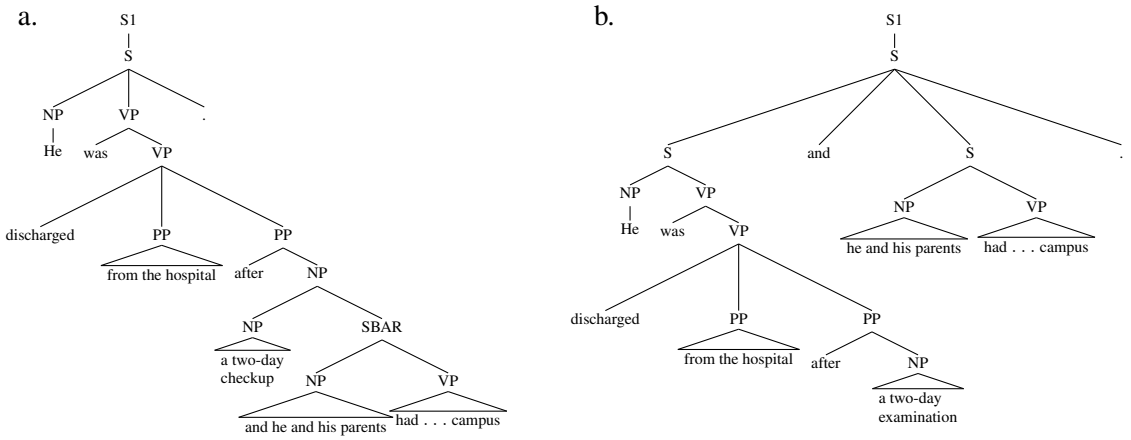

Figure 1: a. Original Parse tree b. Corrected Parse tree after replacement of unknown word *checkup* by *examination*

and transformed sentence. Thus, after annotating the transformed sentence with SRL, we can transfer the roles to the original sentence through this mapping. Transformation functions can be divided into two categories. The first category is *Transformations From List* which uses external resources like WordNet, VerbNet and Word Clusters. The second is *Learned Transformations* that uses transformation rules that have been learned from training data.

4.1 Transformation From List

I. Replacement of Predicate:

As noted in (Huang and Yates, 2010), 6.1% of the predicates in the Brown test set do not appear in WSJ training set and 11.8% appear at most twice. Since the semantic roles of a sentence depend on the predicate, these infrequent predicates hurt SRL performance on new domains. Note that since all predicates in PropBank are verbs, we will use the words *predicate* and *verb* interchangeably.

We count the frequency of each predicate and its accuracy in terms of F1 score over the training data. If the frequency or the F1 score of the predicate in the test sentence is below a threshold, we perturb that predicate. We take all the verbs in the same class of VerbNet[1] as the original verb (in case the verb is present in multiple classes, we take all the classes). In case the verb is not present in VerbNet, we take its synonyms from WordNet. If there is no synonym in WordNet, we take the hypernyms.

From this collection of new verbs, we select verbs that have a high accuracy and a high frequency in training. We replace the original verb with each of these new verbs and generate one new sentence for each new verb; the sentence is retained if the parse score for the new sentence is higher than the parse score for the original sentence.[2] VerbNet has defined a set of verb-independent thematic roles and grouped the verbs according to their usage in frames with identical thematic roles. But PropBank annotation was with respect to each verb. So the same thematic role is labeled as different roles for different verbs in PropBank. For example, both *warn* and *advise* belong to the same VerbNet class (37.9) and take thematic roles of *Recipient* (person being warned or advised) and *Topic* (topic of warning or advising). But *Recipient* was marked as *A2* for *warn* and *A1* for *advise* and *Topic* was marked as *A1* for *warn* and *A2* for *advise* in PropBank annotation. Semlink[3] provides a mapping from the thematic role to PropBank role for each verb. After the SRL annotates the new sentence with PropBank roles for the new verb, we map the PropBank roles of the new verb to their corresponding thematic roles and then map the thematic roles to the corresponding PropBank roles for the original verb.

II. Replacement and Removal of Quoted Strings:

Quoted sentences can vary a lot from one domain to another. For example, in WSJ, quoted sentences are like formal statements but in Brown, these are like informal conversations. We generate the transformations in the following ways:

1) We use the content of the quoted string as one

[1] http://verbs.colorado.edu/ mpalmer/projects/verbnet.html

[2] Parse score is the parse probability returned by Charniak or Stanford parser.

[3] http://verbs.colorado.edu/semlink/

sentence. 2) We replace each quoted string in turn with a simple sentence (*This is good*) to generate a new sentence. 3) If a sentence has a quoted string in the beginning, we move that quoted string after the first NP and VP that immediately follow the quoted string. For example, from the input sentence, *"We just sit quiet", he said.* we generate the sentences 1) *We just sit quiet* 2) *"This is good", he said.* 3) *He said, "We just sit quiet".*

III. Replacement of Unseen Words:

A major difficulty for domain adaptation is that some words in the new domain do not appear in the training domain. In the Brown test set, 5% of total words were never seen in the WSJ training set.

Given an unseen word which is not a verb, we replace it with WordNet synonyms and hypernyms that were seen in the training data. We used the clusters obtained in (Liang, 2005) from running the Brown algorithm (Brown et al., 1992) on Reuters 1996 dataset. But since this cluster was generated automatically, it is noisy. So we chose replacements from the Brown clusters selectively. We only replace those words for which the POS tagger and the syntactic parser predicted different tags. For each such word, we find its cluster and select the set of words from the cluster. We delete from this set all words that do not take at least one part-of-speech tag that the original word can take (from WordNet). For each candidate synonym or hypernym or cluster member, we get a new sentence. Finally we only keep those sentences that have higher parse scores than the original sentence.

IV. Sentence Split based on Stop Symbols:

We split each sentence based on stop symbols like *;* and . . Each of the splitted sentences becomes one transformation of the original sentence.

V. Sentence Simplification:

We have a set of heuristics for simplifying the constituents of the parse tree; for example, replacing an NP with its first and last word, removal of PRN phrases etc. We apply these heuristics and generate simpler sentences until no more simplification is possible. Examples of our heuristics are given in Table 1.

Note that we can use composition of multiple transformation functions as one function. A composition $p_1 \odot p_2(s) = \cup_{a \in p_1(s)} p_2(a)$. We apply II$\odot$I, III$\odot$I, IV$\odot$I and V$\odot$I.

Node	Input Example	Simplified Example	Operation
NP	**He and she** ran.	**He** ran.	replace
NP	**The big man** ran.	**The man** ran.	replace
ADVP	He ran **fast**.	He ran.	delete
PP	He ran **in the field**.	He ran.	delete
PRN	He **– though sick –** ran.	He ran.	delete
VP	He **walked** and ran.	He ran.	delete
TO	I want him **to** run.	I want that he can ran.	rewrite

Table 1: Examples of Simplifications (Predicate is run)

4.2 Learned Transformations

The learned model is inaccurate over verbs and roles that are infrequent in the training data. The purpose of the learned transformation is to transfer such a phrase in the test sentence in place of a phrase of a simpler sentence; this is done such that there exists a mapping from the role of the phrase in the new sentence to the role of the phrase in the original sentence.

Phrase Representation: A phrase tuple is a 3-tuple (t, i, h) where, t is the phrase type, i is the index, and h is the headword of the phrase. We denote by PR the Phrase Representation of a sentence – an ordered list of phrase tuples. A phrase tuple corresponds to a node in the tree. We only consider phrase tuples that correspond to nodes that are (1) a sibling of the predicate node or (2) a sibling of an ancestor of the predicate node. Phrase tuples in PR are sorted based on their position in the sentence. The *index* i of the phrase tuple containing the predicate is taken to be zero with the indices of the phrase tuples on the left (right) sequentially decreasing (increasing).

Transformation Rule: We denote by *Label(n, s)* the semantic role of nth phrase in the PR of the sentence s. Let *Replace*(n_s, n_t, s_s, s_t) be a new sentence that results from inserting the phrase n_s in sentence s_s instead of phrase n_t in sentence s_t. We will refer to s_t as *target sentence* and to n_t as the *target phrase*. Let sp be a sequence of phrase tuples named as *source pattern*. If Label$(n_s, s_s) = r_1$ and Label$(n_t$, Replace$(n_s, n_t, s_s, s_t)) = r_2$, then denote $f(r_2) = r_1$. In this case we call the 6-tuple (s_t, n_t, p, sp, n_s, f) a *transformation rule*. We call f the

label correspondence function.

Example: Consider the sentence s_t = *"But it did not sing."* and the rule τ: $(s_t, n_t, p, sp, n_s, f)$. Let: $n_t = -3$, $p =$ *entitle*,
$sp = [-2, NP, \phi][-1, AUX, \phi][0, V, entitle][1, \phi, to]$
$n_s = -2$, $f = \{<A0, A2>\} \cup \{<Ai, Ai> | i \neq 0\}$.

The PR of $\tau.s_t$ is {[−4, CC, But] [−3, NP, it] [−2, AUX, did] [−1, RB, not] [0, VB, sing] [1, ., .]}. Consider the input sentence s_s: *Mr. X was entitled to a discount* . with PR of {[−2, NP, X] [−1, AUX, was] [0, V, entitle] [1, PP, to][2, ., .]}. Since $\tau.sp$ is a subsequence of the PR of s_s, τ will apply to the predicate *entitle* of s_s. The transformed sentence is:

s_{tr} = Replace$(\tau.n_s, \tau.n_t, s_s, \tau.s_t)$ = *But Mr. X did not sing.* with PR of {[−4, CC, But] [−3, NP, X] [−2, AUX, did] [−1, RB, not] [0, VB, sing] [1, ., .]}. If the SRL system assigns the semantic role of $A0$ to the phrase *Mr. X* of s_{tr}, the semantic role of *Mr. X* in s_s can be recovered through $\tau.f$ since $\tau.f(A0) = A2 =$ Label$(-2, s_s)$.

While checking if $\tau.sp$ is a subsequence of the PR of the input sentence, ϕ in each tuple of $\tau.sp$ has to be considered a trivial match. So τ will match the sentence *He is entitled to a reward.* with PR = {[−2, NP, He] [−1, AUX, is] [0, V, entitle] [1, PP, to][2, ., .]} but will not match the sentence *The conference was entitled a big success.* with PR = {[−2, NP, conference] [−1, AUX, was] [0, V, entitle] [1, S, **success**][2, ., .]} (mismatch position is bolded). The index of a phrase tuple cannot be ϕ, only the head word or type can be ϕ and the rules with more ϕ strings in the source pattern are more general since they can match more sentences.

Algorithm 1 GenerateRules

```
1: Input: predicate v, semantic role r, Training sentences D, SRL Model M
2: Output: set of rules R
3: R ⇐ GetInitialRules(v, r, D, M)
4: repeat
5:     J ⇐ ExpandRules(R)
6:     K ⇐ R ∪ J
7:     sort K based on accuracy, support, size of source pattern
8:     select some rules R ⊂ K based on database coverage
9: until all rules in R have been expanded before
10: return R
```

The algorithm for finding rules for a semantic role r of a predicate v is given in Algorithm 1. It is a specific to general beam search procedure that starts with a set of initial rules (Line 3, detail in Algorithm 2) and finds new rules from these rules (Line 5, detail in Algorithm 3). In Line 7, the rules are sorted by decreasing order of accuracy, support and number of ϕ strings in the source pattern. In Line 8, a set of rules are selected to cover all occurrences of the semantic role r with the predicate v a specific number of times. This process continues until no new rules are found. Note that these rules need to be learned only once and can be used for every new domain.

Algorithm 2 GetInitialRules

```
1: Input: predicate v, semantic role r, Training sentences D, SRL-Model M
2: Output: Set of initial rules I
3: I ⇐ φ
4: T ⇐ {s ∈ D : length(s) <= e}
5: S ⇐ {s ∈ D : s has role r for predicate v}
6: M ⇐ Set of all semantic roles
7: for each phrase p1 in s1 ∈ S with gold label r for predicate v do
8:     for each phrase p2 in s2 ∈ T labeled as a core argument do
9:         if s1 ≠ s2 and p1 and p2 have same phrase types then
10:            τ ⇐ empty rule
11:            τ.st ⇐ s2, τ.p ⇐ v
12:            τ.nt ⇐ index of p2 in PR of s2
13:            τ.ns ⇐ index of p1 in PR of s1
14:            τ.sp ⇐ phrase tuples for phrases from p1 to v and two
               phrases after v in PR of s1
15:            L ⇐ φ
16:            for each sentence s3 ∈ D with predicate v do
17:                if τ.sp is a subsequence of PR of s3 then
18:                    x ⇐ replace(τ.ns, τ.nt, s3, τ.st)
19:                    annotate x with SRL using M
20:                    r1 ⇐ the gold standard semantic role of the
                       phrase with index τ.ns in PR of s3
21:                    r2 ⇐ Label(τ.nt, x)
22:                    if r2 ∉ L then
23:                        insert(r2, r1) in τ.f
24:                        L = L ∪ {r2}
25:                    end if
26:                end if
27:            end for
28:            for each role j ∈ M − L do
29:                insert(j, j) in τ.f
30:            end for
31:            I ⇐ I∪ {τ}
32:        end if
33:    end for
34: end for
35: return I
```

The algorithm for generating initial rules for the semantic role r of predicate v is given in Algorithm 2. Shorter sentences are preferred to be target sentences(Line 4). A rule τ is created for every (p_1,p_2) pair where p_1, p_2 are phrases, p_1 has the semantic role r in some sentence s_1, p_2 is labeled as a core argument($A0 - A5$) in some sentence in T and the phrase types of p_1 and p_2 in their respective parse trees are same(Lines $7 - 9$). Every sentence s_3 in

training corpus with predicate $\tau.p$ is a potential candidate for applying τ(Line 16) if $\tau.sp$ is a subsequence of PR of s_3(Line 17). After applying τ to s_3, a transformed sentence x is created(Line 18). Lines $20-26$ find the semantic role r_2 of the transferred phrase from SRL annotation of x using model M and create a mapping from r_2 to the gold standard role r_1 of the phrase in s_3. L maintains the set of semantic roles for which mappings have been created. In lines $28-30$, all unmapped roles are mapped to themselves.

The algorithm for creating new rules from a set of existing rules is given in Algorithm 3. Lines $4-13$ generate all immediate more general neighbors of the current rule by nullifying the headword or phrase type element in any of the phrase tuples in its source pattern.

Algorithm 3 ExpandRules

```
1:  Input: a set of rules R
2:  Output: a set of expanded rules E
3:  E ⇐ φ
4:  for each phrase tuple c in the source pattern of r ∈ R do
5:      if c is not the tuple for predicate then
6:          create a new rule r′ with all components of r
7:          mark the head word of c in the source pattern of r′ to φ
8:          add r′ to E
9:          create a new rule r′′ with all components of r
10:         mark the phrase type of c in the source pattern of r′′ to φ
11:         add r′′ to E
12:     end if
13: end for
14: return E
```

5 Combination by Joint Inference

The transformation functions transform an input sentence into a set of sentences T. From each transformed sentence t_i, we get a set of argument candidates S_i. Let $S = \bigcup_{i=1}^{|T|} S_i$ be the set of all arguments. Argument classifier assigns scores for each argument over the output labels(roles) in S that is then converted into a probability distribution over the possible labels using the softmax function (Bishop, 1995). Note that multiple arguments with the same span can be generated from multiple transformed sentences.

First, we take all arguments from S with distinct span and put them in S'. For each argument *arg* in S', we calculate scores over possible labels as the sum over the probability distribution (over output labels) of all arguments in S that have the same span as *arg* divided by the number of sentences in T that contained *arg*. This results in a set of arguments with distinct spans and for each argument, a set of scores over possible labels. Following the joint inference procedure in (Punyakanok et al., 2008), we want to select a label for each argument such that the total score is maximized subject to some constraints. Let us index the set S' as $S'^{1:M}$ where $M = |S'|$. Also assume that each argument can take a label from a set P. The set of arguments in $S'^{1:M}$ can take a set of labels $c^{1:M} \in P^{1:M}$. Given some constraints, the resulting solution space is limited to a feasible set F; the inference task is: $c^{1:M} = \arg\max_{c^{1:M} \in F(P^{1:M})} \sum_{i=1}^{M} score(S'^i = c^i)$.

The constraints used are: 1) No overlapping or embedding argument. 2) No duplicate argument for core arguments A0-A5 and AA. 3) For C-arg, there has to be an arg argument.

6 Experimental Setup

In this section, we discuss our experimental setup for the semantic role labeling system. Similar to the CoNLL 2005 shared tasks, we train our system using sections 02-21 of the Wall Street Journal portion of Penn TreeBank labeled with PropBank. We test our system on an annotated Brown corpus consisting of three sections (ck01 - ck03).

Since we need to annotate new sentences with syntactic parse, POS tags and shallow parses, we do not use annotations in the CoNLL distribution; instead, we re-annotate the data using publicly available part of speech tagger and shallow parser[1], Charniak 2005 parser (Charniak and Johnson, 2005) and Stanford parser (Klein and Manning, 2003).

Our baseline SRL model is an implementation of (Punyakanok et al., 2008) which was the top performing system in CoNLL 2005 shared task. Due to space constraints, we omit the details of the system and refer readers to (Punyakanok et al., 2008).

7 Results

Results for ADUT using only the top parse of Charniak and Stanford are shown in Table 2. The Baseline model using top Charniak parse (*BaseLine-Charniak*) and top Stanford parse (*BaseLine-Stanford*) score respectively 76.4 and 73.3 on the

[1] http://cogcomp.cs.illinois.edu/page/software

WSJ test set. Since we are interested in adaptation, we report and compare results for Brown test set only. On this set, both *ADUT-Charniak* and *ADUT-Stanford* significantly outperform their respective baselines. We compare with the state-of-the-art system of (Surdeanu et al., 2007). In (Surdeanu et al., 2007), the authors use three models: Model 1 and 2 do sequential tagging of chunks obtained from shallow parse and full parse. Model 3 assumes each predicate argument maps to one syntactic constituent and classifies it individually. So Model 3 matches our baseline model. *ADUT-Charniak* outperforms the best individual model (Model 2) of (Surdeanu et al., 2007) by 1.6% and Model 3 by 3.9%. We also tested another system that used cluster features and word embedding features computed following (Collobert and Weston, 2008). But we did not see any performance improvement on Brown over baseline.

System	P	R	F1
BaseLine-Charniak	69.6	61.8	65.5
ADUT-Charniak	72.75	66.1	**69.3**
BaseLine-Stanford	70.8	56.5	62.9
ADUT-Stanford	72.5	60.0	65.7
(Surdeanu et al., 2007)(Model 2)	71.8	64.0	67.7
(Surdeanu et al., 2007)(Model 3)	72.4	59.7	65.4

Table 2: Comparing single parse system on Brown.

All state-of-the-art systems for SRL are a combination of multiple systems. So we combined *ADUT-Stanford*, *ADUT-Charniak* and another system *ADUT-Charniak-2* based on 2nd best Charniak parse using joint inference. In Table 3, We compare with (Punyakanok et al., 2008) which was the top performing system in CoNLL 2005 shared task. We also compare with the multi parse system of (Toutanova et al., 2008) which uses a global joint model using multiple parse trees. In (Surdeanu et al., 2007), the authors experimented with several combination strategies. Their first combination strategy was similar to ours where they directly combined the outputs of different systems using constraints (denoted as *Cons* in Table 3). But their best result on Brown set was obtained by treating the combination of multiple systems as a meta-learning problem. They trained a new model to score candidate arguments produced by individual systems before combining them through constraints (denoted as *LBI* in Table 3). We also compare with (Huang and Yates, 2010) where the authors retrained a SRL model using HMM features learned over unlabeled data of WSJ and Brown.

System	P	R	F1	Retrain
(Punyakanok et al., 2008)	73.4	62.9	67.8	×
(Toutanova et al., 2008)	NR	NR	68.8	×
(Surdeanu et al., 2007) (Cons)	78.2	62.1	69.2	×
(Surdeanu et al., 2007) (LBI)	81.8	61.3	70.1	×
ADUT-combined	74.3	67.0	**70.5**	×
(Huang and Yates, 2010)	77.0	70.9	**73.8**	✓

Table 3: Comparison of the multi parse system on Brown.

Table 3 shows that *ADUT-Combined* performs better than *(Surdeanu et al., 2007) (Cons)* when individual systems have been combined similarly. We believe that the techniques in (Surdeanu et al., 2007) of using multiple models of different kinds (two based on sequential tagging of chunks to capture arguments whose boundaries do not match a syntactic constituent) and training an additional model to combine the outputs of individual systems are orthogonal to the performance improvement that we have and applying these methods will further increase the performance of our final system which is a research direction we want to pursue in future.

We did an ablation study to determine which transformations help and by how much. Table 4 presents results when only one transformation is active at a time. We see that each transformation improves over the baseline.

The effect of the transformation of *Replacement of Predicate* on infrequent verbs is shown in Table 5. This transformation improves F1 as much as 6% on infrequent verbs.

The running time for *ADUT-Charniak* on Brown set is 8 hours compared to SRL training time of 20 hours. Average number of transformed sentences generated by *ADUT-Charniak* for every sentence from Brown is 36. The times are calculated based on a machine with 2x 6-Core Xeon X5650 Processor with 48G memory.

Transformation	P	R	F1
Baseline	69.6	61.8	65.5
Replacement of Unknown Words	70.6	62.1	**66.1**
Replacement of Predicate	71.2	62.8	**66.8**
Replacement of Quotes	71.0	63.4	**67.0**
Simplification	70.3	62.9	**66.4**
RuleTransformation	70.9	62.2	**66.2**
Sentence Split	70.8	62.1	**66.2**
Together	72.75	66.1	**69.3**

Table 4: Ablation Study for ADUT-Charniak

Frequency	Baseline	Replacement of Predicate
0	64.2	**67.8**
less than 3	59.7	**65.1**
less than 7	58.9	**64.8**
all predicates	65.5	**66.78**

Table 5: Performance on Infrequent Verbs for the Transformation of *Replacement of Predicate*

8 Related Work

Traditional adaptation techniques like (DauméIII, 2007; Chelba and Acero, 2004; Finkel and Manning, 2009; Jiang and Zhai, 2007; Blitzer et al., 2006; Huang and Yates, 2009; Ando and Zhang, 2005; Ming-wei Chang and Roth, 2010) need to retrain the model for every new domain. In (Umansky-Pesin et al., 2010), there was no retraining; instead, a POS tag was predicted for every unknown word in the new domain by considering contexts of that word collected by web search queries. We differ from them in that our transformations are *label-preserving*; moreover, our transformations aim at making the target text resemble the training text. We also present an algorithm to learn transformation rules from training data. Our application domain, SRL, is also more complex and structured than POS tagging.

In (McClosky et al., 2010), the task of multiple source parser adaptation was introduced. The authors trained parsing models on corpora from different domains and given a new text, used a linear combination of trained models. Their approach requires annotated data from multiple domains as well as unlabeled data for the new domain, which is not needed in our framework. In (Huang and Yates, 2010), the authors trained a HMM over the Brown test set and the WSJ unlabeled data. They derived features from Viterbi optimal states of single words and spans of words and retrained their models using these features. In (Vickrey and Koller, 2008), a large number of hand-written rules were used to simplify the parse trees and reduce syntactic variation to overcome feature sparsity. We have several types of transformations, and use less than 10 simplification heuristics, based on replacing larger phrases with smaller phrases and deleting unnecessary parse tree nodes. There are also some methods for unsupervised semantic role labeling (Swier and Stevenson, 2004), (Abend et al., 2009) that easily adapt across domains but their performances are not comparable to supervised systems.

9 Conclusion

We presented a framework for adaptating natural language *text* so that *models* can be used across domains without modification. Our framework supports adapting to new domains without any data or knowledge of the target domain. We showed that our approach significantly improves SRL performance over the state-of-the-art single parse based system on Brown set. In the future, we would like to extend this approach to other NLP problems and study how combining multiple systems can further improve its performance and robustness.

Acknowledgements This research is sponsored by the Army Research Laboratory (ARL) under agreement W911NF-09-2-0053 and by the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the view of the ARL, the DARPA, AFRL, or the US government.

References

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

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple labeled

and unlabeled data . *Journal of Machine Learning Research*.

Christopher Bishop. 1995. *Neural Networks for Pattern recognition, chapter 6.4: Modelling conditional distributions*. Oxford University Press.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. D. Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling . In *Proceedings of CoNLL*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*.

Ciprian Chelba and Alex Acero. 2004. Little data can help a lot. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

Hal DauméIII. 2007. Frustratingly easy domain adaptation. In *Proceedings of the the Annual Meeting of the Association of Computational Linguistics (ACL)*.

Jenny R. Finkel and Christopher D. Manning. 2009. Hierarchical bayesian domain adaptation . In *Proceedings of NAACL*.

Fei Huang and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence-labeling . In *Proceedings of ACL*.

Fei Huang and Alexander Yates. 2010. Open-domain semantic role labeling by modeling word spans. In *Proceedings of ACL*.

Jing Jiang and ChengXiang Zhai. 2007. Instance weighting for domain adaptation in nlp. In *Proceedings of ACL*.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Proceedings of NIPS*.

Percy Liang. 2005. Semi-supervised learning for natural language. *Masters thesis, Massachusetts Institute of Technology*.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Proceedings of NAACL*.

Michael Connor Ming-wei Chang and Dan Roth. 2010. The necessity of combining adaptation methods. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Massachusetts, USA.

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

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research*, 29:105–151.

Robert S. Swier and Suzanne Stevenson. 2004. Unsupervised semantic role labelling. In *Proceedings of Empirical Methods in Natural Language Processing*.

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

Shulamit Umansky-Pesin, Roi Reichart, and Ari Rappoport. 2010. A multi-domain web-based algorithm for pos tagging of unknown words . In *Proceedings of Coling*.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of the ACL-HLT*.