Discriminative Learning over Constrained Latent Representations

Ming-Wei Chang and **Dan Goldwasser** and **Dan Roth** and **Vivek Srikumar**
University of Illinois at Urbana Champaign
Urbana, IL 61801
{mchang,goldwas1,danr,vsrikum2}@uiuc.edu

Abstract

This paper proposes a general learning framework for a class of problems that require learning over latent intermediate representations. Many natural language processing (NLP) decision problems are defined over an expressive intermediate representation that is not explicit in the input, leaving the algorithm with both the task of recovering a good intermediate representation and learning to classify correctly. Most current systems separate the learning problem into two stages by solving the first step of recovering the intermediate representation heuristically and using it to learn the final classifier. This paper develops a novel joint learning algorithm for both tasks, that uses the final prediction to guide the selection of the best intermediate representation. We evaluate our algorithm on three different NLP tasks – transliteration, paraphrase identification and textual entailment – and show that our joint method significantly improves performance.

1 Introduction

Many NLP tasks can be phrased as decision problems over complex linguistic structures. Successful learning depends on correctly encoding these (often latent) structures as features for the learning system. Tasks such as transliteration discovery (Klementiev and Roth, 2008), recognizing textual entailment (RTE) (Dagan et al., 2006) and paraphrase identification (Dolan et al., 2004) are a few prototypical examples. However, the input to such problems does not specify the latent structures and the problem is defined in terms of surface forms only. Most current solutions transform the raw input into a meaningful intermediate representation[1], and then encode its structural properties as features for the learning algorithm.

Consider the RTE task of identifying whether the meaning of a short text snippet (called the *hypothesis*) can be inferred from that of another snippet (called the *text*). A common solution (MacCartney et al., 2008; Roth et al., 2009) is to begin by defining an alignment over the corresponding entities, predicates and their arguments as an intermediate representation. A classifier is then trained using features extracted from the intermediate representation. The idea of using a intermediate representation also occurs frequently in other NLP tasks (Bergsma and Kondrak, 2007; Qiu et al., 2006).

While the importance of finding a good intermediate representation is clear, emphasis is typically placed on the later stage of extracting features over this intermediate representation, thus separating learning into **two stages** – specifying the latent representation, and then extracting features for learning. The latent representation is obtained by an inference process using predefined models or well-designed heuristics. While these approaches often perform well, they ignore a useful resource when generating the latent structure – the labeled data for the final learning task. As we will show in this paper, this results in degraded performance for the actual classification task at hand. Several works have considered this issue (McCallum et al., 2005; Goldwasser and Roth, 2008b; Chang et al., 2009; Das and Smith, 2009); however, they provide solutions

[1] In this paper, the phrases "intermediate representation" and "latent representation" are used interchangeably.

Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL, pages 429–437,
Los Angeles, California, June 2010. ©2010 Association for Computational Linguistics

that do not easily generalize to new tasks.

In this paper, we propose a unified solution to the problem of learning to make the classification decision **jointly** with determining the intermediate representation. Our *Learning Constrained Latent Representations (***LCLR***) framework* is guided by the intuition that there is *no* intrinsically good intermediate representation, but rather that a representation is good only to the extent to which it improves performance on the final classification task. In the rest of this section we discuss the properties of our framework and highlight its contributions.

Learning over Latent Representations This paper formulates the problem of *learning over latent representations* and presents a novel and general solution applicable to a wide range of NLP applications. We analyze the properties of our learning solution, thus allowing new research to take advantage of a well understood learning and optimization framework rather than an ad-hoc solution. We show the generality of our framework by successfully applying it to three domains: transliteration, RTE and paraphrase identification.

Joint Learning Algorithm In contrast to most existing approaches that employ domain specific heuristics to construct intermediate representations to learn the final classifier, our algorithm learns to construct the optimal intermediate representation to support the learning problem. Learning to represent is a difficult structured learning problem however, unlike other works that use labeled data at the intermediate level, our algorithm only uses the binary supervision supplied for the final learning problem.

Flexible Inference Successful learning depends on constraining the intermediate representation with task-specific knowledge. Our framework uses the declarative Integer Linear Programming (ILP) inference formulation, which makes it easy to define the intermediate representation and to inject knowledge in the form of *constraints*. While ILP has been applied to structured output learning, to the best of our knowledge, this is the first work that makes use of ILP in formalizing the general problem of learning intermediate representations.

2 Preliminaries

We introduce notation using the Paraphrase Identification task as a running example. This is the binary classification task of identifying whether one sentence is a paraphrase of another. A paraphrase pair from the MSR Paraphrase corpus (Quirk et al., 2004) is shown in Figure 1. In order to identify that the sentences paraphrase each other , we need to align constituents of these sentences. One possible alignment is shown in the figure, in which the dotted edges correspond to the aligned constituents. An alignment can be specified using binary variables corresponding to every edge between constituents, indicating whether the edge is included in the alignment. Different activations of these variables induce the space of *intermediate representations.*

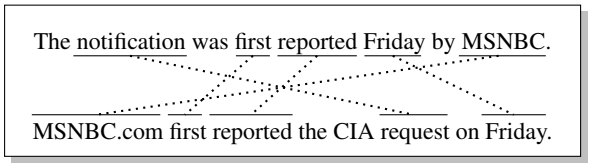

Figure 1: The dotted lines represent a possible intermediate representation for the paraphrase identification task. Since different representation choices will impact the binary identification decision directly, our approach chooses the representation that facilitates the binary learning task.

To formalize this setting, let $\mathbf{x}$ denote the input to a decision function, which maps $\mathbf{x}$ to $\{-1, 1\}$. We consider problems where this decision depends on an intermediate representation (for example, the collection of all dotted edges in Figure 1), which can be represented by a binary vector $\mathbf{h}$.

In the literature, a common approach is to separate the problem into two stages. First, a generation stage predicts $\mathbf{h}$ for each $\mathbf{x}$ using a pre-defined model or a heuristic. This is followed by a learning stage, in which the classifier is trained using $\mathbf{h}$. In our example, if the generation stage predicts the alignment shown, then the learning stage would use the features computed based on the alignments. Formally, the two-stage approach uses a pre-defined inference procedure that finds an intermediate representation $\mathbf{h}'$. Using features $\Phi(\mathbf{x}, \mathbf{h}')$ and a learned weight vector θ, the example is classified as positive if $\theta^T \Phi(\mathbf{x}, \mathbf{h}') \geq 0$.

However, in the two stage approach, the latent representation, which is provided to the learning algorithm, is determined before learning starts, and without any feedback from the final task. It is dictated by the intuition of the developer. This approach makes two implicit assumptions: first, it assumes

the existence of a "correct" latent representation and, second, that the model or heuristic used to generate it is the correct one for the learning problem at hand.

3 Joint Learning with an Intermediate Representation

In contrast to two-stage approaches, we use the annotated data for the final classification task to learn a suitable intermediate representation which, in turn, helps the final classification.

Choosing a good representation is an optimization problem that selects which of the elements (features) of the representation best contribute to successful classification given some legitimacy constraints; therefore, we (1) set up the optimization framework that finds legitimate representations (Section 3.1), and (2) learn an objective function for this optimization problem, such that it makes the best final decision (Section 3.2.)

3.1 Inference

Our goal is to correctly predict the final label rather than matching a "gold" intermediate representation. In our framework, attempting to learn the final decision drives both the selection of the intermediate representation and the final predictions.

For each $\mathbf{x}$, let $\Gamma(\mathbf{x})$ be the set of all substructures of all possible intermediate representations. In Figure 1, this could be the set of all alignment edges connecting the constituents of the sentences. Given a vocabulary of such structures of size N, we denote intermediate representations by $\mathbf{h} \in \{0,1\}^N$, which "select" the components of the vocabulary that constitute the intermediate representation. We define $\phi_s(\mathbf{x})$ to be a feature vector over the substructure s, which is used to describe the characteristics of s, and define a weight vector $\mathbf{u}$ over these features.

Let $\mathcal{C}$ denote the set of feasible intermediate representations $\mathbf{h}$, specified by means of linear constraints over $\mathbf{h}$. While $\Gamma(\mathbf{x})$ might be large, the set of those elements in $\mathbf{h}$ that are active can be constrained by controlling $\mathcal{C}$. After we have learned a weight vector $\mathbf{u}$ that scores intermediate representations for the final classification task, we define our decision function as

$$f_{\mathbf{u}}(\mathbf{x}) = \max_{\mathbf{h}\in\mathcal{C}} \mathbf{u}^T \sum_{s\in\Gamma(\mathbf{x})} h_s \phi_s(\mathbf{x}), \qquad (1)$$

and classify the input as positive if $f_{\mathbf{u}}(\mathbf{x}) \geq 0$.

In Eq. (1), $\mathbf{u}^T\phi_s(\mathbf{x})$ is the score associated with the substructure s, and $f_{\mathbf{u}}(\mathbf{x})$ is the score for the entire intermediate representation. Therefore, our decision function $f_{\mathbf{u}}(\mathbf{x}) \geq 0$ makes use of the intermediate representation and its score to classify the input. An input is labeled as positive if its underlying intermediate structure allows it to cross the decision threshold. The intermediate representation is chosen to maximize the overall score of the input. This design is especially beneficial for many phenomena in NLP, where only positive examples have a meaningful underlying structure. In our paraphrase identification example, good alignments generally exist only for positive examples.

One unique feature of our framework is that we treat Eq. (1) as an Integer Linear Programming (ILP) instance. A concrete instantiation of this setting to the paraphrase identification problem, along with the actual ILP formulation is shown in Section 4.

3.2 Learning

We now present an algorithm that learns the weight vector $\mathbf{u}$. For a loss function $\ell : \mathbb{R} \to \mathbb{R}$, the goal of learning is to solve the following optimization problem:

$$\min_{\mathbf{u}} \frac{\lambda}{2}\|\mathbf{u}\|^2 + \sum_i \ell\left(-y_i f_{\mathbf{u}}(\mathbf{x}_i)\right) \qquad (2)$$

Here, λ is the regularization parameter. Substituting Eq. (1) into Eq. (2), we get

$$\min_{\mathbf{u}} \frac{\lambda}{2}\|\mathbf{u}\|^2 + \sum_i \ell\left(-y_i \max_{\mathbf{h}\in\mathcal{C}} \mathbf{u}^T \sum_{s\in\Gamma(\mathbf{x})} h_s \phi_s(\mathbf{x}_i)\right) \qquad (3)$$

Note that there is a maximization term inside the global minimization problem, making Eq. (3) a non-convex problem. The minimization drives $\mathbf{u}$ towards smaller empirical loss while the maximization uses $\mathbf{u}$ to find the best representation for each example.

The algorithm for Learning over Constrained Latent Representations (LCLR) is listed in Algorithm 1. In each iteration, first, we find the best feature representations for all positive examples (lines 3-5). This step can be solved with an off-the-shelf ILP solver. Having fixed the representations for the positive examples, we update the $\mathbf{u}$ by solving Eq. (4) at line 6 in the algorithm. It is important to observe

Algorithm 1 *LCLR* :The algorithm that optimizes (3)

1: initialize: $\mathbf{u} \leftarrow \mathbf{u}_0$
2: **repeat**
3: **for all** positive examples $(\mathbf{x}_i, y_i = 1)$ **do**
4: Find $\mathbf{h}_i^* \leftarrow \arg\max_{\mathbf{h} \in \mathcal{C}} \sum_s h_s \mathbf{u}^T \phi_s(\mathbf{x}_i)$
5: **end for**
6: Update $\mathbf{u}$ by solving

$$\min_{\mathbf{u}} \frac{\lambda}{2}\|\mathbf{u}\|^2 + \sum_{i:y_i=1} \ell(-\mathbf{u}^T \sum_s h_{i,s}^* \phi_s(\mathbf{x}_i)) + \sum_{i:y_i=-1} \ell(\max_{\mathbf{h} \in \mathcal{C}} \mathbf{u}^T \sum_s h_s \phi_s(\mathbf{x}_i)) \quad (4)$$

7: **until** convergence
8: return $\mathbf{u}$

Algorithm 2 Cutting plane algorithm to optimize Eq. (4)

1: for each negative example x_j, $H_j \leftarrow \emptyset$
2: **repeat**
3: **for** each negative example x_j **do**
4: Find $\mathbf{h}_j^* \leftarrow \arg\max_{\mathbf{h} \in \mathcal{C}} \sum_s h_s \mathbf{u}^T \phi_s(\mathbf{x}_j)$
5: $H_j \leftarrow H_j \cup \{\mathbf{h}_j^*\}$
6: **end for**
7: Solve

$$\min_{\mathbf{u}} \frac{\lambda}{2}\|\mathbf{u}\|^2 + \sum_{i:y_i=1} \ell(-\mathbf{u}^T \sum_s h_{i,s}^* \phi_s(\mathbf{x}_i)) + \sum_{i:y_i=-1} \ell(\max_{\mathbf{h} \in H_j} \mathbf{u}^T \sum_s h_s \phi_s(\mathbf{x}_i)) \quad (5)$$

8: **until** no new element is added to any H_j
9: return $\mathbf{u}$

that for positive examples in Eq. (4), we use the intermediate representations $\mathbf{h}^*$ from line 4.

Algorithm 1 satisfies the following property:

Theorem 1 *If the loss function ℓ is a non-decreasing function, then the objective function value of Eq.* (3) *is guaranteed to decrease in every iteration of Algorithm 1. Moreover, if the loss function is also convex, then Eq.* (4) *in Algorithm 1 is convex.*

Due to the space limitation, we omit the proof.

Theoretically, we can use any loss function that satisfies the conditions of the theorem. In the experiments in this paper, we use the squared-hinge loss function: $\ell(-yf_{\mathbf{u}}(\mathbf{x})) = \max(0, 1 - yf_{\mathbf{u}}(\mathbf{x}))^2$.

Recall that Eq. (4) is not the traditional SVM or logistic regression formulation. This is because inside the inner loop, the best representation for each negative example must be found. Therefore, we need to perform inference for every negative example when updating the weight vector solution. Instead of solving a difficult non-convex optimization problem (Eq. (3)), LCLR iteratively solves a series of easier problems (Eq. (4)). This is especially true for our loss function because Eq. (4) is convex and can be solved efficiently.

We use a cutting plane algorithm to solve Eq. (4). A similar idea has been proposed in (Joachims et al., 2009). The algorithm for solving Eq. (4) is presented as Algorithm 2. This algorithm uses a "cache" H_j to store all intermediate representations for negative examples that have been seen in previous iterations (lines 3-6) [2]. The difference between Eq. (5) in line 7 of Algorithm 2 and Eq. (4) is that in Eq. (5), we do not search over the entire space of intermediate representations. The search space for the minimization problem Eq. (5) is restricted to the cache H_j. Therefore, instead of solving the minimization problem Eq. (4), we can now solve several simpler problems shown in Eq. (5). The algorithm is guaranteed to stop (line 8) because the space of intermediate representations is finite. Furthermore, in practice, the algorithm needs to consider only a small subset of "hard" examples before it converges.

Inspired by (Hsieh et al., 2008), we apply an efficient coordinate descent algorithm for the *dual* formulation of (5) which is guaranteed to find its global minimum. Due to space considerations, we do not present the derivation of dual formulation and the details of the optimization algorithm.

4 Encoding with ILP: A Paraphrase Identification Example

In this section, we define the latent representation for the paraphrase identification task. Unlike the earlier example, where we considered the alignment of lexical items, we describe a more complex intermediate representation by aligning graphs created using semantic resources.

An input example is represented as two acyclic

[2] In our implementation, we keep a global cache H_j for each negative example x_j. Therefore, in Algorithm 2, we start with a non-empty cache improving the speed significantly.

graphs, G_1 and G_2, corresponding to the first and second input sentences. Each vertex in the graph contains word information (lemma and part-of-speech) and the edges denote dependency relations, generated by the Stanford parser (Klein and Manning, 2003). The intermediate representation for this task can now be defined as an alignment between the graphs, which captures lexical and syntactic correlations between the sentences.

We use $V(G)$ and $E(G)$ to denote the set of vertices and edges in G respectively, and define four hidden variable types to encode vertex and edge mappings between G_1 and G_2.

- The **word-mapping** variables, denoted by h_{v_1,v_2}, define possible pairings of vertices, where $v_1 \in V(G_1)$ and $v_2 \in V(G_2)$.
- The **edge-mapping** variables, denoted by h_{e_1,e_2}, define possible pairings of the graphs edges, where $e_1 \in E(G_1)$ and $e_2 \in E(G_2)$.
- The **word-deletion** variables $h_{v_1,*}$ (or h_{*,v_2}) allow for vertices $v_1 \in V(G_1)$ (or $v_2 \in V(G_2)$) to be deleted. This accounts for omission of words (like function words).
- The **edge-deletion** variables, $h_{e_1,*}$ (or h_{*,e_2}) allow for deletion of edges from G_1 (or G_2).

Our inference problem is to find the optimal set of hidden variable activations, restricted according to the following set of linear constraints

- Each vertex in G_1 (or G_2) can either be mapped to a single vertex in G_2 (or G_1) or marked as deleted. In terms of the **word-mapping** and **word-deletion** variables, we have

$$\forall v_1 \in V(G_1); h_{v_1,*} + \sum_{v_2 \in V(G_2)} h_{v_1,v_2} = 1 \quad (6)$$

$$\forall v_2 \in V(G_2); h_{*,v_2} + \sum_{v_1 \in V(G_1)} h_{v_1,v_2} = 1 \quad (7)$$

- Each edge in G_1 (or G_2) can either be mapped to a single edge in G_2 (or G_1) or marked as deleted. In terms of the **edge-mapping** and **edge-deletion** variables, we have

$$\forall e_1 \in E(G_1); h_{e_1,*} + \sum_{e_2 \in E(G_2)} h_{e_1,e_2} = 1 \quad (8)$$

$$\forall e_2 \in E(G_2); h_{*,e_2} + \sum_{e_1 \in E(G_1)} h_{e_1,e_2} = 1 \quad (9)$$

- The edge mappings can be active if, and only if, the corresponding node mappings are active. Suppose $e_1 = (v_1, v_1') \in E(G_1)$ and $e_2 = (v_2, v_2') \in E(G_2)$, where $v_1, v_1' \in V(G_1)$ and $v_2, v_2' \in V(G_2)$. Then, we have

$$h_{v_1,v_2} + h_{v_1',v_2'} - h_{e_1,e_2} \leq 1 \quad (10)$$

$$h_{v_1,v_2} \geq h_{e_1,e_2}; h_{v_1',v_2'} \geq h_{e_1,e_2} \quad (11)$$

These constraints define the feasible set for the inference problem specified in Equation (1). This inference problem can be formulated as an ILP problem with the objective function from Equation (1):

$$\max_{\mathbf{h}} \quad \sum_s h_s \mathbf{u}^T \phi_s(\mathbf{x})$$
$$\text{subject to} \quad (6)\text{-}(11); \ \forall s; h_s \in \{0,1\} \quad (12)$$

This example demonstrates the use of integer linear programming to define intermediate representations incorporating domain intuition.

5 Experiments

We applied our framework to three different NLP tasks: transliteration discovery (Klementiev and Roth, 2008), RTE (Dagan et al., 2006), and paraphrase identification (Dolan et al., 2004).

Our experiments are designed to answer the following research question: "Given a binary classification problem defined over latent representations, will the joint LCLR algorithm perform better than a two-stage approach?" To ensure a fair comparison, both systems use the same feature functions and definition of intermediate representation. We use the same ILP formulation in both configurations, with a single exception – the objective function parameters: the two stage approach uses a task-specific heuristic, while LCLR learns it iteratively.

The ILP formulation results in very strong two stage systems. For example, in the paraphrase identification task, even our two stage system is the current state-of-the-art performance. In these settings, the improvement obtained by our joint approach is non-trivial and can be clearly attributed to the superiority of the joint learning algorithm. Interestingly, we find that our more general approach is better than specially designed joint approaches (Goldwasser and Roth, 2008b; Das and Smith, 2009).

Since the objective function (3) of the joint approach is not convex, a good initialization is required. We use the weight vector learned by the two

stage approach as the starting point for the joint approach. The algorithm terminates when the relative improvement of the objective is smaller than 10^{-5}.

5.1 Transliteration Discovery

Transliteration discovery is the problem of identifying if a word pair, possibly written using two different character sets, refers to the same underlying entity. The intermediate representation consists of all possible character mappings between the two character sets. Identifying this mapping is not easy, as most writing systems do not perfectly align phonetically and orthographically; rather, this mapping can be context-dependent and ambiguous.

For an input pair of words (w_1, w_2), the intermediate structure $\mathbf{h}$ is a mapping between their characters, with the latent variable h_{ij} indicating if the i^{th} character in w_1 is aligned to the j^{th} character in w_2. The feature vector associated with the variable h_{ij} contains unigram character mapping, bigram character mapping (by considering surrounding characters). We adopt the *one-to-one mapping* and *non-crossing* constraint used in (Chang et al., 2009).

We evaluated our system using the English-Hebrew corpus (Goldwasser and Roth, 2008a), which consists of 250 positive transliteration pairs for training, and 300 pairs for testing. As negative examples for training, we sample 10% from random pairings of words from the positive data. We report two evaluation measurements – (1) the Mean Reciprocal Rank (MRR), which is the average of the multiplicative inverse of the rank of the correct answer, and (2) the accuracy (Acc), which is the percentage of the top rank candidates being correct.

We initialized the two stage inference process as detailed in (Chang et al., 2009) using a Romanization table to assign uniform weights to prominent character mappings. This initialization procedure resembles the approach used in (Bergsma and Kondrak, 2007). An alignment is first built by solving the constrained optimization problem. Then, a support vector machine with squared-hinge loss function is used to train a classifier using features extracted from the alignment. We refer to this two stage approach as *Alignment+Learning*.

The results summarized in Table 1 show the significant improvement obtained by the joint approach (95.4% MRR) compared to the two stage approach (85.7%). Moreover, LCLR outperforms the joint system introduced in (Goldwasser and Roth, 2008b).

Transliteration System	**Acc**	**MRR**
(Goldwasser and Roth, 2008b)	N/A	89.4
Alignment + Learning	80.0	85.7
LCLR	**92.3**	**95.4**

Table 1: Experimental results for transliteration. We compare a *two-stage* system: "Alignment+Learning" with LCLR, our *joint* algorithm. Both "Alignment+Learning" and LCLR use the same features and the same intermediate representation definition.

5.2 Textual Entailment

Recognizing Textual Entailment (RTE) is an important textual inference task of predicting if a given *text* snippet, entails the meaning of another (the *hypothesis*). In many current RTE systems, the entailment decision depends on successfully aligning the constituents of the text and hypothesis, accounting for the internal linguistic structure of the input.

The raw input – the text and hypothesis – are represented as directed acyclic graphs, where vertices correspond to words. Directed edges link verbs to the head words of semantic role labeling arguments produced by (Punyakanok et al., 2008). All other words are connected by dependency edges. The intermediate representation is an alignment between the nodes and edges of the graphs. We used three hidden variable types from Section 4 – **word-mapping**, **word-deletion** and **edge-mapping**, along with the associated constraints as defined earlier. Since the text is typically much longer than the hypothesis, we create **word-deletion** latent variables (and features) only for the hypothesis.

The second column of Table 2 lists the resources used to generate features corresponding to each hidden variable type. For word-mapping variables, the features include a WordNet based metric (WNSim), indicators for the POS tags and negation identifiers. We used the state-of-the-art coreference resolution system of (Bengtson and Roth, 2008) to identify the canonical entities for pronouns and extract features accordingly. For word deletion, we use only the POS tags of the corresponding tokens (generated by the LBJ POS tagger[3]) to generate features. For edge

[3] http://L2R.cs.uiuc.edu/~cogcomp/software.php

Hidden Variable	RTE features	Paraphrase features
word-mapping	WordNet, POS, Coref, Neg	WordNet, POS, NE, ED
word-deletion	POS	POS, NE
edge-mapping	NODE-INFO	NODE-INFO, DEP
edge-deletion	N/A	DEP

Table 2: Summary of latent variables and feature resources for the entailment and paraphrase identification tasks. See Section 4 for an explanation of the hidden variable types. The linguistic resources used to generate features are abbreviated as follows – POS: Part of speech, Coref: Canonical coreferent entities; NE: Named Entity, ED: Edit distance, Neg: Negation markers, DEP: Dependency labels, NODE-INFO: corresponding node alignment resources, N/A: Hidden variable not used.

Entailment System	Acc
Median of TAC 2009 systems	61.5
Alignment + Learning	65.0
LCLR	**66.8**

Table 3: Experimental results for recognizing textual entailment. The first row is the median of best performing systems of all teams that participated in the RTE5 challenge (Bentivogli et al., 2009). *Alignment + Learning* is our two-stage system implementation, and LCLR is our joint learning algorithm. Details about these systems are provided in the text.

mapping variables, we include the features of the corresponding word mapping variables, scaled by the word similarity of the words forming the edge.

We evaluated our system using the RTE-5 data (Bentivogli et al., 2009), consisting of 600 sentence pairs for training and testing respectively, in which positive and negative examples are equally distributed. In these experiments the joint LCLR algorithm converged after 5 iterations.

For the two stage system, we used WN-Sim to score alignments during inference. The word-based scores influence the edge variables via the constraints. This two-stage system (the Alignment+Learning system) is significantly better than the median performance of the RTE-5 submissions. Using LCLR further improves the result by almost 2%, a substantial improvement in this domain.

5.3 Paraphrase Identification

Our final task is Paraphrase Identification, discussed in detail at Section 4. We use all the four hidden variable types described in that section. The features used are similar to those described earlier for the RTE system and are summarized in Table 2.

Paraphrase System	Acc
Experiments using (Dolan et al., 2004)	
(Qiu et al., 2006)	72.00
(Das and Smith, 2009)	73.86
(Wan et al., 2006)	75.60
Alignment + Learning	76.23
LCLR	**76.41**
Experiments using **Extended data set**	
Alignment + Learning	72.00
LCLR	**72.75**

Table 4: Experimental Result For Paraphrasing Identification. Our joint LCLR approach achieves the best results compared to several previously published systems, and our own two stage system implementation (*Alignment + Learning*). We evaluated the systems performance across two datasets: (Dolan et al., 2004) dataset and the Extended dataset, see the text for details. Note that LCLR outperforms (Das and Smith, 2009), which is a specifically designed joint approach for this task.

We used the MSR paraphrase dataset of (Dolan et al., 2004) for empirical evaluation. Additionally, we generated a second corpus (called the *Extended dataset*) by sampling 500 sentence pairs from the MSR dataset for training and using the entire test collection of the original dataset. In the Extended dataset, for every sentence pair, we extended the longer sentence by concatenating it with itself. This results in a more difficult inference problem because it allows more mappings between words. Note that the performance on the original dataset sets the ceiling on the second one.

The results are summarized in Table 4. The first part of the table compares the LCLR system with a two stage system (*Alignment + Learning*) and three published results that use the MSR dataset. (We only list single systems in the table[4]) Interestingly, although still outperformed by our joint LCLR algorithm, the two stage system is able perform significantly better than existing systems for that dataset (Qiu et al., 2006; Das and Smith, 2009; Wan et al., 2006). We attribute this improvement, consistent across both the ILP based systems, to the intermediate representation we defined.

We hypothesize that the similarity in performance between the joint LCLR algorithm and the two stage

[4]Previous work (Das and Smith, 2009) has shown that combining the results of several systems improves performance.

(*Alignment + Learning*) systems is due to the limited intermediate representation space for input pairs in this dataset. We evaluated these systems on the more difficult *Extended* dataset. Results indeed show that the margin between the two systems increases as the inference problem becomes harder.

6 Related Work

Recent NLP research has largely focused on two-stage approaches. Examples include RTE (Zanzotto and Moschitti, 2006; MacCartney et al., 2008; Roth et al., 2009); string matching (Bergsma and Kondrak, 2007); transliteration (Klementiev and Roth, 2008); and paraphrase identification (Qiu et al., 2006; Wan et al., 2006).

(MacCartney et al., 2008) considered constructing a latent representation to be an independent task and used manually labeled *alignment* data (Brockett, 2007) to tune the inference procedure parameters. While this method identifies alignments well, it does not improve entailment decisions. This strengthens our intuition that the latent representation should be guided by the final task.

There are several exceptions to the two-stage approach in the NLP community (Haghighi et al., 2005; McCallum et al., 2005; Goldwasser and Roth, 2008b; Das and Smith, 2009); however, the intermediate representation and the inference for constructing it are closely coupled with the application task. In contrast, LCLR provides a general formulation that allows the use of expressive constraints, making it applicable to many NLP tasks.

Unlike other latent variable SVM frameworks (Felzenszwalb et al., 2009; Yu and Joachims, 2009) which often use task-specific inference procedure, LCLR utilizes the declarative inference framework that allows using *constraints* over intermediate representation and provides a general platform for a wide range of NLP tasks.

The optimization procedure in this work and (Felzenszwalb et al., 2009) are quite different. We use the coordinate descent and cutting-plane methods ensuring we have fewer parameters and the inference procedure can be easily parallelized. Our procedure also allows different loss functions. (Cherry and Quirk, 2008) adopts the Latent SVM algorithm to define a language model. Unfortunately, their implementation is not guaranteed to converge.

In CRF-like models with latent variables (McCallum et al., 2005), the decision function marginalizes over the all hidden states when presented with an input example. Unfortunately, the computational cost of applying their framework is prohibitive with constrained latent representations. In contrast, our framework requires only the *best* hidden representation instead of marginalizing over all possible representations, thus reducing the computational effort.

7 Conclusion

We consider the problem of learning over an intermediate representation. We assume the existence of a latent structure in the input, relevant to the learning problem, but not accessible to the learning algorithm. Many NLP tasks fall into these settings and each can consider a different hidden input structure. We propose a unifying thread for the different problems and present a novel framework for Learning over Constrained Latent Representations (LCLR). Our framework can be applied to many different latent representations such as parse trees, orthographic mapping and tree alignments. Our approach contrasts with existing work in which learning is done over a *fixed* representation, as we advocate *jointly* learning it with the final task.

We successfully apply the proposed framework to three learning tasks – Transliteration, Textual Entailment and Paraphrase Identification. Our joint LCLR algorithm achieves superior performance in all three tasks. We attribute the performance improvement to our novel training algorithm and flexible inference procedure, allowing us to encode domain knowledge. This presents an interesting line of future work in which more linguistic intuitions can be encoded into the learning problem. For these reasons, we believe that our framework provides an important step forward in understanding the problem of learning over hidden structured inputs.

Acknowledgment We thank James Clarke and Mark Sammons for their insightful comments. This research was partly sponsored by the Army Research Laboratory (ARL) (accomplished under Cooperative Agreement Number W911NF-09-2-0053) and by Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the ARL or of AFRL.

References

E. Bengtson and D. Roth. 2008. Understanding the value of features for coreference resolution. In *EMNLP*.

L. Bentivogli, I. Dagan, H. T. Dang, D. Giampiccolo, and B. Magnini. 2009. The fifth PASCAL recognizing textual entailment challenge. In *Proc. of TAC Workshop*.

S. Bergsma and G. Kondrak. 2007. Alignment-based discriminative string similarity. In *ACL*.

C. Brockett. 2007. Aligning the RTE 2006 corpus. In *Technical Report MSR-TR-2007-77, Microsoft Research*.

M. Chang, D. Goldwasser, D. Roth, and Y. Tu. 2009. Unsupervised constraint driven learning for transliteration discovery. In *NAACL*.

C. Cherry and C. Quirk. 2008. Discriminative, syntactic language modeling through latent svms. In *Proc. of the Eighth Conference of AMTA*.

I. Dagan, O. Glickman, and B. Magnini, editors. 2006. *The PASCAL Recognising Textual Entailment Challenge.*

D. Das and N. A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *ACL*.

W. Dolan, C. Quirk, and C. Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING*.

P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. 2009. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

D. Goldwasser and D. Roth. 2008a. Active sample selection for named entity transliteration. In *ACL*. Short Paper.

D. Goldwasser and D. Roth. 2008b. Transliteration as constrained optimization. In *EMNLP*.

A. Haghighi, A. Ng, and C. Manning. 2005. Robust textual inference via graph matching. In *HLT-EMNLP*.

C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear svm. In *ICML*.

T. Joachims, T. Finley, and Chun-Nam Yu. 2009. Cutting-plane training of structural svms. *Machine Learning*.

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

A. Klementiev and D. Roth. 2008. Named entity transliteration and discovery in multilingual corpora. In Cyril Goutte, Nicola Cancedda, Marc Dymetman, and George Foster, editors, *Learning Machine Translation*.

B. MacCartney, M. Galley, and C. D. Manning. 2008. A phrase-based alignment model for natural language inference. In *EMNLP*.

A. McCallum, K. Bellare, and F. Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *UAI*.

V. Punyakanok, D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*.

L. Qiu, M.-Y. Kan, and T.-S. Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *EMNLP*.

C. Quirk, C. Brockett, and W. Dolan. 2004. Monolingual machine translation for paraphrase generation. In *EMNLP*.

D. Roth, M. Sammons, and V.G. Vydiswaran. 2009. A framework for entailed relation recognition. In *ACL*.

S. Wan, M. Dras, R. Dale, and C. Paris. 2006. Using dependency-based features to take the ẗpara-farceöut of paraphrase. In *Proc. of the Australasian Language Technology Workshop (ALTW)*.

C. Yu and T. Joachims. 2009. Learning structural svms with latent variables. In *ICML*.

F. M. Zanzotto and A. Moschitti. 2006. Automatic learning of textual entailments with cross-pair similarities. In *ACL*.