Transliteration as Constrained Optimization

Dan Goldwasser **Dan Roth**
Department of Computer Science
University of Illinois
Urbana, IL 61801
{goldwas1,danr}@uiuc.edu

Abstract

This paper introduces a new method for identifying named-entity (NE) transliterations in bilingual corpora. Recent works have shown the advantage of discriminative approaches to transliteration: given two strings (w_s, w_t) in the source and target language, a classifier is trained to determine if w_t is the transliteration of w_s. This paper shows that the transliteration problem can be formulated as a constrained optimization problem and thus take into account contextual dependencies and constraints among character bi-grams in the two strings. We further explore several methods for learning the objective function of the optimization problem and show the advantage of learning it discriminately. Our experiments show that the new framework results in over 50% improvement in translating English NEs to Hebrew.

1 Introduction

Named entity (NE) transliteration is the process of transcribing a NE from a source language to some target language based on phonetic similarity between the entities. Identifying transliteration pairs is an important component in many linguistic applications which require identifying out-of-vocabulary words, such as machine translation and multilingual information retrieval (Klementiev and Roth, 2006b; Hermjakob et al., 2008).

It may appear at first glance that identifying the phonetic correlation between names based on an orthographic analysis is a simple, straight-forward task; however in many cases a consistent deterministic mapping between characters does not exist; rather, the mapping depends on the context the characters appear in and on transliteration conventions which may change across domains. Figure 1 exhibits two examples of NE transliterations in English and Hebrew, with the correct mapping across the two scripts. Although the two Hebrew names share a common prefix[1], this prefix can be mapped into a single English character or into two different characters depending on the context it appears in. Similarly, depending on the context it appears in, the English character a can be mapped into different characters or to an "empty" character.

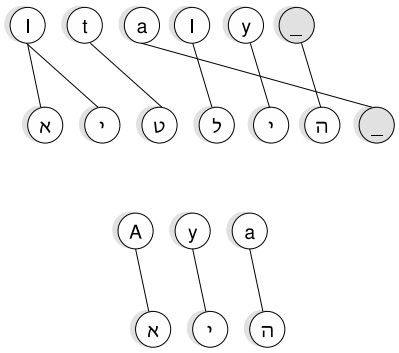

Figure 1: Named entities transliteration pairs in English and Hebrew and the character level mapping between the two names. The Hebrew names can be romanized as *ee-ta-l-ya* and *a-ya*

[1] In all our example the Hebrew script is shown left-to-right to simplify the visualization of the transliteration mapping.

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, pages 353–362,
Honolulu, October 2008.

In recent years, as it became clear that solutions that are based on linguistics rules are not satisfactory, machine learning approaches have been developed to address this problem. The common approach adopted is therefore to view this problem as a classification problem (Klementiev and Roth, 2006a; Tao et al., 2006) and train a discriminative classifier. That is, given two strings, one in the source and the other in the target language, extract pairwise features, and train a classifier that determines if one is a transliteration of the other. Several papers have followed up on this basic approach and focused on semi-supervised approaches to this problem or on extracting better features for the discriminative classifier (Klementiev and Roth, 2006b; Bergsma and Kondrak, 2007; Goldwasser and Roth, 2008). While it has been clear that the relevancy of pairwise features is context sensitive and that there are contextual constraints among them, the hope was that a discriminative approach will be sufficient to account for those by weighing features appropriately. This has been shown to be difficult for language pairs which are very different, such as English and Hebrew (Goldwasser and Roth, 2008).

In this paper, we address these difficulties by proposing to view the transliteration decision as a globally phrased constrained optimization problem. We formalize it as an optimization problem over a set of local pairwise features – character n-gram matches across the two string – and subject to legitimacy constraints.

We use a discriminatively trained classifier as a way to learn the objective function for the global constrained optimization problem. Our technical approach follows a large body of work developed over the last few years, following (Roth and Yih, 2004) that has formalized global decisions problems in NLP as constrained optimization problems and solved these optimization problems using Integer Linear Programming (ILP) or other methods (Punyakanok et al., 2005; Barzilay and Lapata, 2006; Clarke and Lapata, ; Marciniak and Strube, 2005).

We investigate several ways to train our objective function, which is represented as a dot product between a set of features chosen to represent a pair (w_s, w_t), and a vector of initial weights. Our first baseline makes use of all features extracted from a pair, along with a simple counting method to determine initial weights. We then use a method similar to (Klementiev and Roth, 2006a; Goldwasser and Roth, 2008) in order to discriminatively train a better weight vector for the objective function.

Our key contribution is that we use a constrained optimization approach also to determine a better feature representation for a given pair. (Bergsma and Kondrak, 2007) attempted a related approach to restricting the set of features representing a transliteration candidate. However, rather than directly aligning the two strings as done there, we exploit the expressiveness of the ILP formulation and constraints to generate a better representation of a pair. This is the representation we then use to discriminatively learn a better weight vector for the objective function used in our final model.

Our experiments focus on Hebrew-English transliteration, which were shown to be very difficult in a previous work (Goldwasser and Roth, 2008). We show very significant improvements over existing work with the same data set, proving the advantage of viewing the transliteration decision as a global inference problem. Furthermore, we show the importance of using a discriminatively trained objective function.

The rest of the paper is organized as follows. The main algorithmic contribution of this paper is described in Sec. 2. Our experimental study is describes in Sec. 3 and Sec. 4 concludes.

2 Using inference for transliteration

In this section we present our transliteration decision framework, which is based on solving a constrained optimization problem with an objective function that is discriminatively learned. Our framework consists of three key elements:

1. **Decision Model** When presented with a NE in the source language w_s and a set of candidates $\{w_t\}_1^k$ in the target language, the decision model ranks the candidate pairs (w_s, w_t) and selects the "best" candidate pair. This is framed as an optimization problem

$$w_t^* = argmax_i\{w \cdot F(w_s, w_t^i)\}, \quad (1)$$

where F is a feature vector *representation* of the pair (w_s, w_t^i) and w is a vector of *weights* assigned to each feature.

2. **Representation** A pair $s = (w_s, w_t)$ of source and target NEs is represented as a vector of *features*, each of which is a pair of character n-grams, from w_s and w_t, resp. Starting with a baseline representation introduced in (Klementiev and Roth, 2006a), denoted here $AF(s)$, we refine this representation to take into account dependencies among the individual n-gram pairs. This refinement process is framed as a constrained optimization problem:

$$F(s)^* = argmax_{F \subseteq AF}\{w \cdot AF(s)\}, \quad (2)$$

subject to a set C of linear constraints. Here AF is the initial representation (All−Features), w is a vector of *weights* assigned to each feature and C is a set of constraints accounting for interdependencies among features.

3. **Weight Vector** Each pairwise n-gram feature is associated with a weight; this weigh vector is used in both optimization formulations above. The weight vector is determined by considering the whole training corpus. The initial weight vector is obtained generatively, by counting the relative occurrence of substring pairs in positive examples. The representation is refined by discriminatively training a classifier to maximize transliteration performance on the training data. In doing that, each example is represented using the feature vector representation described above.

The three key operations described above are being used in several stages, with different parameters (weight vectors and representations) as described in Alg. 1. In each stage a different element is refined. The input to this process is a training corpus Tr=(D_S,D_T) consisting of NE transliteration pairs $s = (w_s, w_t)$, where w_s, w_t are NEs in the source and target language, respectively. Each such sample point is initially represented as a feature vector $AF(s)$ (for All−Features), where features are pairs of substrings from the two words (following (Klementiev and Roth, 2006a)).

Given the set of feature vectors generated by applying AF to Tr, we assign initial weights W to the features ((1) in Alg. 1). These weights form the initial objective function used to construct a new feature based representation, Informative−Features, $IF_W(s)$ ((2) in Alg. 1). Specifically, for an instance s, $IF_W(s)$ is the solution of the optimization problem in Eq. 2, with W as the weight vector, $AF(s)$ as the representation, and a set of constraints ensuring the "legitimacy" of the selected set of features (Sec. 2.2.1).

Input: Training Corpora Tr=(D_S,D_T)
Output: Transliteration model $\mathcal{M}$

1. Initial Representation and Weights

For each sample s ∈ Tr, use AF to generate a feature vector
$\{(f_s, f_t)^1, (f_s, f_t)^2, \ldots, (f_s, f_t)^n\} \in \{0,1\}^n$.
Define W:$f \rightarrow \mathcal{R}$ s.t. foreach feature f=(f_s, f_t)

$$W(f) = \frac{\#(f_s,f_t)}{\#(f_s)} \times \frac{\#(f_s,f_t)}{\#(f_t)}$$

2. Inferring Informative Representation (W)

Modify the initial representation by solving the following constrained optimization problem:
$IF_W(s)^* = argmax_{IF(s) \subseteq (AF(s))} W \cdot AF(s)$,
subject to constraints C.

3. Discriminative Training

Train a discriminative model on Tr, using $\{IF(s)\}_{s \in Tr}$.
Let W_D be the new weight vector obtained by discriminative training.

4. Inferring Informative Representation (W_D)

Modify the initial representation by solving the following constrained optimization problem. This time, the objective function is determined by the discriminatively trained weight vector W_D.
$IF_{W_D}(s)^* = argmax_{IF(s) \subseteq (AF(s))} W_D \cdot AF(s)$,
subject to constraints C.

5. Decision Model

Given a word w_s and a list of candidates $w_t^1, w_t^2, \ldots w_t^k$, the chosen transliteration is w_{t^*}, determined by:

$$t^* = argmax_i\{W_D \cdot IF_{W_D}((w_s, w_t^i))\}$$

Algorithm 1: Transliteration Framework.

The new feature extraction operator $IF_W(s)$ is now used to construct a new representation of the training corpus. With this representation, we train discriminately a new weight vector W_D. This weight vector, now defines a new objective function for the optimization problem in Eq. 2; W_D is the weight vector and $AF(s)$ the representation. We de-

note by $IF_{W_D}(s)$ the solution of this optimization problem for an instance s.

Given a representation and a weight vector, the optimization problem in Eq. 1 is used to find the transliteration of w_s. Our best decision model makes use of Eq. 1 using W_D as the feature vector and $IF_{W_D}(s)$ as the feature representation of s.

The rest of this section provides details on the operations and how we use them in different stages.

2.1 Initial Representation and Weights

The feature space we consider consists of n potential features, each feature $f = (f_s, f_t)$ represents a pairing of character level n-grams, where $f_s \in \{$*Source-Language* $\cup$ *empty-string* $\}$ and $f_t \in \{$*Target-Language* $\cup$ *empty-string*$\}$. A given sample (w_s, w_t) consisting of a pair of NEs is represented as a features vector $s \in \{0,1\}^n$. We say that a feature f^i is active if $f^i = 1$ and that $s_1 \subset s_2$, $\iff \{f^i\}_{\{f^i=1\ in\ s_1\}} \subset \{f^i\}_{\{f^i=1\ in\ s_2\}}$. We represent the active features corresponding to a pair as a bipartite graph $G = (V, E)$, in which each vertex $v \in V$ either represents the empty string, a single character or a bi-gram. V^S, V^T denote the vertices representing source and target language n-grams respectively. Each of these sets is composed of two disjoint subsets: $V_S = V_U^S \cup V_B^S$, $V_T = V_U^T \cup V_B^T$ consisting of vertices representing the uni-gram and bi-gram strings. Given a vertex v, *degree(v, V')*denotes the degree of v in a subgraph of G, consisting only of $V' \subset V$; *index(v)* is the index of the substring represented by v in the original string.

Edges in the bipartite graph represent active features. The only deviation is that the vertex representing the empty string can be connected to any other (non-empty) vertex.

Our initial feature extraction method follows the one presented in (Klementiev and Roth, 2006a), in which the feature space consists of n-gram pairs from the two languages. Given a pair, each word is decomposed into a set of character substrings of up to a given length (including the empty string). Features are generated by pairing substrings from the two sets whose relative positions in the original words differ by k or less places, or formally:

$$E = \{e = (v_i, v_j) \mid (v_i \in V_S \wedge v_j \in V_T) \Rightarrow (index(v_j) + k \geq index(v_i) \geq index(v_j) - k) \wedge (v_i \neq v_{empty-string} \vee v_j \neq v_{empty-string})\}.$$

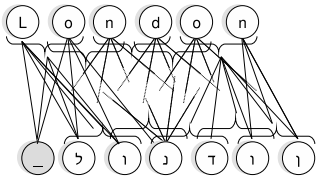

Figure 2: All possible unigram and bigram pairs generated by the AF operator. The Hebrew name can be romanized as *lo-n-do-n*

In our experiments we used *k*=1 which tested empirically, achieved the best performance.

Figure 2 exhibits the active features in the example using the graph representation. We refer to this feature extraction method as $All\text{-}Features$ (AF), and define it formally as an operator $AF : s \rightarrow \{(f_s, f_t)^i\}$ that maps a sample point $s = (w_s, w_t)$ to a set of active features.

The initial sample representation generates features by coupling substrings from the two terms without considering the dependencies between the possible consistent combinations. Ideally, given a positive sample, it is desirable that paired substrings would encode phonetic similarity or a distinctive context in which the two substrings correlate. However, AF simply pairs substrings from the two words, resulting in a noisy representation of the sample point. Given enough positive samples, we assume that features appearing with distinctive frequency will encode the desired relation. We use this observation, and construct a weight vector, associating each feature with a positive number indicating its relative occurrence frequency in the training data representation formed by AF. This weight is computed as follows:

Definition 1 (Initial Feature Weights Vector) *Let* W*:*f $\rightarrow \mathcal{R}$ *s.t. for each feature* f=$\{f_s, f_t\}$*,*

$$W(\text{f}) = \frac{\#(f_s, f_t)}{\#(f_s)} \times \frac{\#(f_s, f_t)}{\#(f_t)},$$

where $\#(f_s, f_t)$ is the number of occurrences of that feature in the positive sample set, and $\#(f_L), L = \{s, t\}$ is the number of occurrences of an individual substring, in any of the features extracted from positive samples in the training set.

These weights transform every example into a weighted graph, where each edge is associated by W with the weight assigned to the feature it represents. As we empirically tested, this initialization assigns high weights to features that preserve the phonetic correlation between the two languages. The top part of figure 5 presents several examples of weights assigned by W to features composed of different English and Hebrew substrings combinations. It can be observed that combination which are phonetically similar are associated with a higher weight. However, as it turns out, transliteration mappings do not consist of "clean" and consistent mappings of phonetically similar substrings. In the following section we explain how to use these weights to generate a more compact representation of samples.

2.2 Inferring Informative Representations

In this section we suggest a new feature extraction method for determining the representation of a given word pair. We use the strength of the active features computed above, along with legitimacy constraints on mappings between source and target strings to find an optimal set of consistent active features that represents a pair. This problem can be naturally encoded as a linear optimization problem, which seeks to maximize a linear objective function determined by W, over a set of variables representing the active features selection, subject to a set of linear constraints representing the dependencies between selections. We follow the formulation given by (Roth and Yih, 2004), and define it as an Integer Linear Programming (ILP) optimization problem, in which each integer variable $a_{(j,k)}$, defined over $\{0, 1\}$, represents whether a feature pairing an n-gram $j \in S$ with an n-gram $k \in T$, is active. Although using ILP is in general NP-hard, it has been used efficiently in many natural language (see section 1). Our experience as well has been that this process is very efficient due to the sparsity of the constraints used.

2.2.1 Constraining Feature Dependencies

To limit the selection of active features in each sample we require that each element in the decomposition of w_s into bi-grams should be paired with an element in w_t, and the vice-versa. We restrict the possible pairs by allowing only a single n-gram to be matched to any other n-gram, with one exception - we allow every bi-gram to be mapped into an empty string. Viewed as a bipartite graph, we allow each node (with the exception of the empty string) to have only one connected edge. These constraints, given the right objective function, should enforce an alignment of bi-grams according to phonetic similarity; for example, the word pairs described in Figure 1, depicts a character level alignment between the words, where in some cases a bi-gram is mapped into a single character and in other cases single characters are mapped to each other, based on phonetic similarity encoded by the two scripts. However, imposing these constraints over the entire set of candidate features would be too restrictive; it is unlikely that one can consistently represent a single "correct" phonetic mapping. We wish to represent both the character level and bi-gram mapping between names as both represent informative features on the correspondence between the names over the two scripts. To allow this, we decompose the problem into two disjoint sets of constraints imposing 1-1 mappings, one over the set of single character substrings and the other over the bi-gram substrings. Given the bipartite graph generated by *AF*, we impose the following constraints:

Definition 2 (Transliteration Constraints) *Let C be the set of constraints, consisting of the following predicates:*

$$\begin{gathered}\forall v \in V^S, \text{ degree(v,}V^S \cup V_U^T) \leq 1 \wedge \\ \forall v \in V^S, \text{ degree(v,}V^S \cup V_B^T) \leq 1 \wedge \\ \forall v \in V^T, \text{ degree(v,}V^T \cup V_U^S) \leq 1 \wedge \\ \forall v \in V^T, \text{ degree(v,}V^T \cup V_B^S) \leq 1\end{gathered}$$

For example, Figure 2 shows the graph of all possible candidates produced by AF. In Figure 3, the graph is decomposed into two graphs, each depicting possible matches between the character level uni-gram or bi-gram substrings. the ILP constraints ensure that in each graph, every node (with the exception of the empty string) has a degree of one . Figure 4 gives the results of the ILP process – a unified graph in which every node has only a single edge associated with it.

Definition 3 (Informative Feature Extraction (IF)) *We define the Informative-Features(IF) feature extraction operator, $IF : s \rightarrow \{(f_s, f_t)^i\}$ as the solution to the ILP problem in Eq. 2. Namely,*

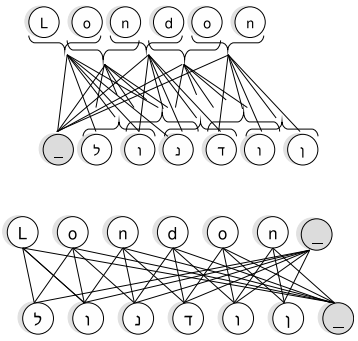

Figure 3: Find informative features by solving an ILP problem. Dependencies between matching decisions are modeled by allowing every node to be connected to a single edge (except the node representing the empty-string).

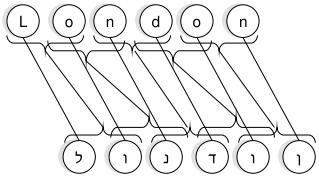

Figure 4: The result of applying the IF operator by solving an ILP problem, represented as a pruned graph.

$$IF(s)^* = argmax_{IF(s) \subseteq (AF(s))} w \cdot AF(s),$$

subject to constraints C.

We will use this operator with $w = W$, defined above, and denote it IF_W, and also use it with a different weight vector, trained discriminatively, as described next.

2.3 Discriminative Training

Using the IF_W operator, we generate a better representation of the training data, which is now used to train a discriminative model. We use a linear classifier trained with a regularized average perceptron update rule (Grove and Roth, 2001) as implemented in SNoW, (Roth, 1998). This learning algorithm provides a simple and general linear classifier that has been demonstrated to work well in other NLP classification tasks, e.g. (Punyakanok et al., 2005), and allows us to incorporate extensions such as strength of features naturally into the training algorithm. We augment each sample in the training data with feature weights; given a sample, the learner is presented with a real-valued feature vector instead of a binary vector. This can be viewed as providing a better starting point for the learner, which improves the learning rate (Golding and Roth, 1999; Ng and Jordan, 2001).

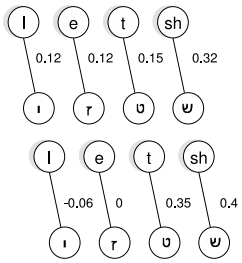

Figure 5: Several examples of weights assigned to features generated by coupling English and Hebrew substrings. Top figure: initial weights. Bottom figure: Discriminatively learned weights. The Hebrew characters, ordered left to right, can be romanized as *y,z,t,sh*

The weight vector learned by the discriminative training is denoted W_D. Given the new weight vector, we can define a new feature extraction operator, that we get by applying the objective function in Eq. 2 with W_D instead of W. Given a sample s, the feature representation generated by this new information extraction operator is denoted $IF_{W_D}(s)$. The key difference between W and W_D is that the latter was trained over a corpora containing both negative and positive examples, and as a result W_D contains negative weights. To increase the impact of training we multiplied the negative weights by 2.

Figure 5 presents some examples of the benefit of discriminately learning the objective function; the weighted edges in the top figure show the values assigned to features by W, while the bottom figure shows the weights assigned by W_D. In all cases, phonetically similar characters were assigned higher scores by W_D, and character pairs not phonetically similar were typically assigned negative weights. It is also interesting to note a special phenomena occurring in English-Hebrew transliterations. The English vowels will be paired to almost any Hebrew character when generating pairs using AF, since vowels in most cases are omitted in Hebrew, there is no distinctive context in which English vowels appear. We can see for example, in the top graph

presented in Figure 5 an edge matching a vowel to a Hebrew character with a high weight, the bottom graph showing the results of the discriminative training process show that this edge is associated with a zero weight score.

2.4 Decision Models

This section defines several transliteration decision models given a word w_s and a list of candidates $w_t^1, w_t^2, \ldots w_t^k$. The models are used to identify the correct transliteration pair from the set of candidates $\{s_i = (w_s, w_t^i)\}_{i=1\ldots k}$.

In all cases, the decision is formulated as in Eq. 1, where different models differ by the representations and weight vectors used.

Decision Model 1 *Ranking the transliteration candidates is done by evaluating*

$$s^* = argmax_i \ W \cdot AF(s_i),$$

which selects the transliteration pair which maximizes the objective function based on the generatively computed weight vector.

Decision Model 2 *Ranking the transliteration candidates is done by evaluating:*

$$s^* = argmax_i \ W_D \cdot AF(s_i)).$$

This decision model is essentially equivalent to the transliteration models used in (Klementiev and Roth, 2006a; Goldwasser and Roth, 2008), in which a linear transliteration model was trained using a feature extraction method equivalent to *AF*.

Decision Model 3 *Ranking the transliteration candidates is done by evaluating:*

$$s^* = argmax_i \ W \cdot IF_W(s_i),$$

which maximizes the objective function with the generatively computed weight vector and the informative feature representation derived based on it.

Decision Model 4 *Ranking the transliteration candidates is done by evaluating:*

$$s^* = argmax_i \ W_D \cdot IF_W(s_i)),$$

which conceptually resembles the transliteration model presented in (Bergsma and Kondrak, 2007), in that a discriminative classifier was trained and used over a pruned feature set.

Decision Model 5 *Ranking the transliteration candidates is done by evaluating:*

$$s^* = argmax_i \ W_D \cdot IF_{W_D}(s_i),$$

which maximize the objective function with the discriminately derived weight vector and the informative features inferred based on it. This decision model is the only model that incorporates discriminative weights as part of the feature extraction process; W_D is used as the objective function used when inferring IF_{W_D}.

3 Evaluation

We evaluated our approach over a corpus of 300 English-Hebrew transliteration pairs, and used another 250 different samples for training the models. We constructed the test set by pairing each English name with all Hebrew names in the corpus. The system was evaluated on its ability to correctly identify the 300 transliteration pairs out of all the possible transliteration candidates. We measured performance using the *Mean Reciprocal Rank* (MRR) measure. This measure, originally introduced in the field of information retrieval, is used to evaluate systems that rank several options according to their probability of correctness. MRR is a natural measure in our settings and has been used previously for evaluating transliteration systems, for example by (Tao et al., 2006).

Given a set Q of queries and their respective responses ranked according to the system's confidence, we denote the rank of the correct response to a query $q_i \in Q$ as $rank(q_i)$. MRR is then defined as the average of the multiplicative inverse of the rank of the correct answer, that is:

$$MRR = \frac{1}{|Q|} \sum_{i=1\ldots|Q|} \frac{1}{rank(q_i)}.$$

In our experiments we solved an ILP problem for every transliteration candidate pairs, and computed MRR with respect to the confidence of our decision model across the candidates. Although this required solving thousands of ILP instances, it posed no computational burden as these instances typically contained a small number of variables and constraints. The entire test set is solved in less than 20 minutes

using the publicly available GLPK package (`http://www.gnu.org/software/glpk/`).

The performance of the different models is summarized in table 1, these results are based on a training set of 250 samples used to train the discriminative transliteration models and also to construct the initial weight vector W. Figure 6 shows performance over different number of training examples. Our evaluation is concerns with the core transliteration and decision models presented here and does not consider any data set optimizations that were introduced in previous works, which we view as orthogonal additions, hence the difference with the results published in (Goldwasser and Roth, 2008).

The results clearly show that our final model, model 5, outperform other models. Interestingly, model 1, a simplistic model, significantly outperforms the discriminative model presented in (Klementiev and Roth, 2006b). We believe that this is due to two reasons. It shows that discriminative training over the representation obtained using AF is not efficient; moreover, this phenomenon is accentuated given that we train over a very small data set, which favors generative estimation of weights. This is also clear when comparing the performance of model 1 to model 4, which shows that learning over the representation obtained using constrained optimization (IF) results in a very significant performance improvement.

The improvement of using IF_W is not automatic. Model 3, which uses IF_W, and model 1, which uses AF, converge to nearly the same result. Both these models use generative weights to make the transliteration decision, and this highlights the importance of discriminative training. Both model 4 and model 5 use discriminatively trained weights and significantly outperform model 3. These results indicate that using constraint optimization to generate the examples' representation in itself may not help; the objective function used in this inference has a significant role in improved performance.

The benefit of discriminatively training the objective function becomes even clearer when comparing the performance of model 5 to that of model 4, which uses the original weight vector when inferring the sample representation.

It can be assumed that this algorithm can benefit from further iterations – generating a new feature representations, training a model on it, and using the resulting model as a new objective function. However, it turns out that after a single round, improved weights due to additional training do not change the feature representation; the inference process does not yield a different outcome.

Decision Model	MRR
Baseline model, used in (KR'06,GR'08)	
Model 2	0.51
Models presented in this paper	
Model 1	0.713
Model 3	0.715
Model 4	0.832
Model 5	0.848

Table 1: Results of the different transliteration models, trained using 250 samples. To facilitate readability (Klementiev and Roth, 2006b; Goldwasser and Roth, 2008) are referenced as KR'06 and GR'08 respectively.

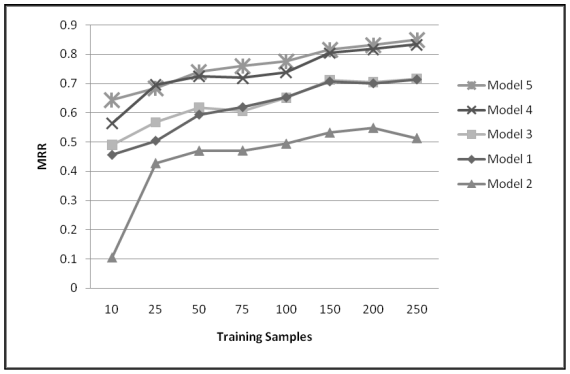

Figure 6: Results of the different constraint optimization transliteration models. Performance is compared relative to the number of samples used for training.

3.1 Normalized Objective Function

Formulating the transliteration decision as an optimization problem also allows us to naturally encode other considerations into our objective function. in this case we give preference to matching short words. We encode this preference as a normalization factor for the objective function. When evaluating on pair (w_s, w_t), we divide the weight vector length of the shorter word; our decision model now becomes:

Decision Model 6 (Model 5 - LengthNormalization)

Decision Model	MRR
Model 5	0.848
Model 5 - LN	0.894

Table 2: Results of using model 5 with and without a normalized objective function. Both models were trained using 250 samples. The LN suffix in the model's name indicate that the objective function used length normalization.

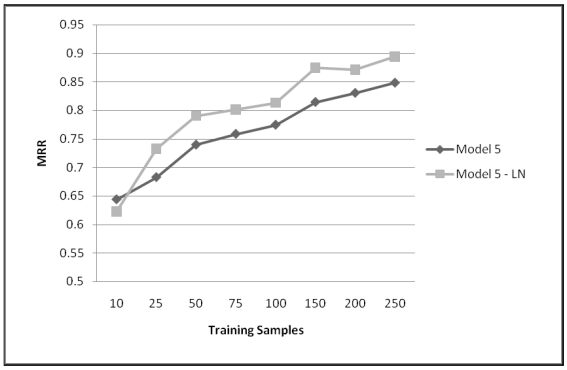

Figure 7: Results of using model 5 with and without a normalized objective function. Performance is compared relative to the number of samples used for training.

Ranking the transliteration candidates is done by evaluating:

$$s^* = argmax_i \ W_D \cdot IF_{W_D}(s_i)/min(|w_s|, |w_t|)$$

As described in table 2 and figure 7, using length normalization significantly improves the results. This can be attributed to the fact that typically Hebrew names are shorter and therefore every pair (w_s, w_t) considered by our model will be effected differently by this normalization factor.

4 Discussion

We introduced a new approach for identifying NE transliteration, viewing the transliteration decision as a global inference problem. We explored several methods for combining discriminative learning in a global constraint optimization framework and showed that discriminatively learning the objective function improves performance significantly.

From an algorithmic perspective, our key contribution is the introduction of a new method, in which learning and inference are used in an integrated way. We use learning to generate an objective function for the inference process; use the inference process to generate a better representation for the learning process, and iterate these stages.

From the transliteration perspective, our key contribution is in deriving and showing the significance of a good representation for a pair of NEs. Our representation captures both phonetic similarity and distinctive occurrence patterns across character level matchings of the two input strings, while enforcing the constraints induced by the interdependencies of the individual matchings. As we show, this representation serves to improve the ability of a discriminative learning algorithm to weigh features appropriately and results in significantly better transliteration models. This representation can be viewed as a compromise between models that do not consider dependencies between local decisions and those that try to align the two strings. Achieving this compromise is one of the advantages of the flexibility allowed by the constrained optimization framework we use. We plan to investigate using more constraints within this framework, such as soft constraints which can penalize unlikely local decisions while not completely eliminating the entire solution.

Acknowledgments

We wish to thank Alex Klementiev and the anonymous reviewers for their insightful comments. This work is partly supported by NSF grant SoD-HCER-0613885 and DARPA funding under the Bootstrap Learning Program.

References

R. Barzilay and M. Lapata. 2006. Aggregation via Set Partitioning for Natural Language Generation. In *Proceedings of HLT/NAACL*, pages 359–366, New York City, USA, June. Association for Computational Linguistics.

S. Bergsma and G. Kondrak. 2007. Alignment-based discriminative string similarity. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 656–663, Prague, Czech Republic, June. Association for Computational Linguistics.

J. Clarke and M. Lapata. Modeling compression with discourse constraints. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Lan-*

guage Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 1–11.

A. R. Golding and D. Roth. 1999. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

D. Goldwasser and D. Roth. 2008. Active sample selection for named entity transliteration. In *Proceedings of ACL-08: HLT, Short Papers*, Columbus, OH, USA, Jun. Association for Computational Linguistics.

A. Grove and D. Roth. 2001. Linear concepts and hidden variables. *Machine Learning*, 42(1/2):123–141.

Ulf Hermjakob, Kevin Knight, and Hal Daumé III. 2008. Name translation in statistical machine translation - learning when to transliterate. In *Proceedings of ACL-08: HLT*, pages 389–397, Columbus, Ohio, June. Association for Computational Linguistics.

A. Klementiev and D. Roth. 2006a. Named entity transliteration and discovery from multilingual comparable corpora. In *Proc. of the Annual Meeting of the North American Association of Computational Linguistics (NAACL)*, pages 82–88, June.

A. Klementiev and D. Roth. 2006b. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proc. of the Annual Meeting of the ACL*, July.

T. Marciniak and M. Strube. 2005. Beyond the Pipeline: Discrete Optimization in NLP. In *Proceedings of the Ninth CoNLL*, pages 136–143, Ann Arbor, Michigan, June. Association for Computational Linguistics.

A. Y. Ng and M. I. Jordan. 2001. On discriminative vs. generative classifiers: A comparison of logistic regression and naïve bayes. In *Neural Information Processing Systems*, pages 841–848.

V. Punyakanok, D. Roth, and W. Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1117–1123.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In Hwee Tou Ng and Ellen Riloff, editors, *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8. Association for Computational Linguistics.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 806–813.

Tao Tao, Su-Youn Yoon, Andrew Fister, Richard Sproat, and ChengXiang Zhai. 2006. Unsupervised named entity transliteration using temporal and phonetic correlation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 250–257, Sydney, Australia, July. Association for Computational Linguistics.