1
Introduction

Machine learning (ML) is a very active research field whose main objective is to learn a prediction function from a given set of examples. In the last decades, ML has been successfully applied in many fields, such as natural language processing, information extraction, computer vision, and computational biology. There are several learning paradigms, but in this work we focus on *supervised* machine learning. In this case, the training examples comprise inputs and their correct outputs.

A classic ML problem is binary classification, in which the prediction output is binary. Here, the learned model discriminates between two classes of examples. However, many important problems involve the prediction of a structure. In these cases, the prediction output comprises many variables with complex interdependencies. Natural language processing (NLP) includes many *structure learning* (SL) problems, such as dependency parsing (DP), part-of-speech (POS) tagging, quotation extraction and coreference resolution. Dependency parsing is to identify a *tree* underlying a given sentence. In POS tagging, for a given input sentence, the prediction output is a *sequence* of tags. In quotation extraction, an input document is *segmented* into non-overlapping quotes that, additionally, are associated with their authors. Given a document with mentions to entities from the real world – like people, companies and places – coreference resolution consists in *clustering* mentions that are references to the same entity.

1.1
Ad Hoc Approaches

Many approaches to solve structured problems are based on complex ML systems that combine several basic classifiers. In order to consider the interdependencies among output variables, the basic classifiers are trained and applied by *ad hoc* strategies that pass information from one classifier to another during training and enforce constraints on the prediction outputs of the basic classifiers. One of the most basic structured problems is multiclass classification. This is a generalization of the binary classification problem

where one needs to discriminate among $K > 2$ classes of interest. There are two common approaches to perform this task by decomposing it on independent binary classification problems. The *one-vs-all* approach trains K binary classifiers, where the k-th classifier, for $k \in \{1, \ldots, K\}$, is trained to discriminate the instances of class k from instances of all other classes. To classify an unseen example, all K classifiers are applied and a rule is used to choose only one class. Another approach to multiclass classification is the *one-vs-one* technique. In this approach, $K(K-1)/2$ binary classifiers are trained in order to discriminate between each pair of classes. To classify an unseen example, a voting scheme is used to enforce the unique-class constraint.

RelHunter (Fernandes et al., 2010b,c) is a general method for relation extraction from text that is based on task decomposition and entropy-guided transformation learning (ETL) (Milidiú et al., 2008; dos Santos and Milidiú, 2009b). Besides the fact that ETL considers complex output dependencies, it is tailored for sequential outputs and thus can not directly consider arbitrary structures. In quotation extraction, for instance, RelHunter trains two ETL models to identify tokens that start or end a quote. Then, another ETL model is trained to discriminate which pairs of start-end tokens are correct quotes and, additionally, to associate them with their authors. A task specific heuristic is later applied to discard overlapping quotes. In Fernandes et al. (2010c), RelHunter is further applied to text chunking, clause identification, hedge detection, and dependency parsing.

Dependency parsing (DP) (Buchholz and Marsi, 2006) is to identify the words that syntactically modify other words in a given sentence. This dependency structure corresponds to a rooted tree whose nodes are the sentence words. In Figure 1.1, we show a dependency tree example. As usual,

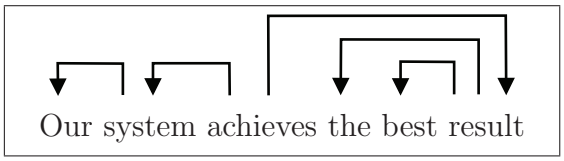

Figure 1.1: Dependency tree example.

in this example, the root node is the main verb of the sentence. An edge (i, j) connects the i-th token to the j-th token and indicates that the latter modifies the former. Additionally, the i-th token is called the *head* token and the j-th token is called the *modifier*. Nivre et al. (2006) propose a dependency parser that relies on a left-to-right deterministic parsing algorithm. Four support vector machines (SVM) are trained to predict parsing actions given features that represent the parser history. Previous word predictions are incorporated in order to consider the interdependency between predictions. Milidiú et al.

(2009) cast DP as a token classification task by using a specific tagging style that provides good generalization. In this tagging style, a modifier token class uniquely identifies its head token. A modifier tag is given by the concatenation of three values: the head token POS tag; how many tokens with the same POS of the head there are between the modifier and its head; and if the head is to the left or to the right of the modifier token. Then, an ETL model is directly trained on the given sentences, along with their features, to predict these specific token tags. This approach achieves good performance but has limitations to generalize on complex sentences with several clauses, since these instances can involve arbitrarily long distances, even with this relative distance.

These ad hoc approaches are developed on a per-task basis and do not provide a general neither principled design pattern. Thus, they are not directly generalized to arbitrary structures and, moreover, most of them have no theoretical guarantees regarding their performances.

1.2 Linear Discriminative Models

In this work, we are interested in *linear discriminative* methods that, regarding binary classification, for instance, learn the parameters of the following linear discriminant function

$$s(\boldsymbol{x}; \boldsymbol{w}) = \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}) \rangle = \sum_{m=1,\ldots,M} w_m \cdot \phi_m(\boldsymbol{x}), \tag{1-1}$$

where $\boldsymbol{\Phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_M(\boldsymbol{x}))$ is a vector of M real-valued feature functions, or simply *features*, that represents the input $\boldsymbol{x}$; $\boldsymbol{w} = (w_1, \ldots, w_M)$ is the parameter vector – also called *model*, which is estimated from examples; and $\langle \cdot, \cdot \rangle$ is the scalar product operator. Then, the prediction function for a given model $\boldsymbol{w}$ is simply

$$F(\boldsymbol{x}; \boldsymbol{w}) = \begin{cases} +1 & \text{if } s(\boldsymbol{x}; \boldsymbol{w}) \geq 0 \\ -1 & \text{otherwise.} \end{cases} \tag{1-2}$$

The learning problem is then to estimate $\boldsymbol{w}$ from a training set $\mathcal{D} = \{(\boldsymbol{x}, y)\}$ comprising correct input-output pairs $(\boldsymbol{x}, y)$, such that $\boldsymbol{x}$ is an input and $y \in \{-1, +1\}$ is the corresponding binary output.

There are plenty of training algorithms for binary classification that follow the *empirical risk minimization* (ERM) principle (Vapnik, 1998). The empirical risk of a model $\boldsymbol{w}$ on the training set $\mathcal{D}$ is given by

$$\mathcal{R}(\mathcal{D}, \boldsymbol{w}) = \sum_{(\boldsymbol{x},y)\in\mathcal{D}} \mathbf{1}[y \neq F(\boldsymbol{x}; \boldsymbol{w})], \tag{1-3}$$

where $\mathbf{1}[p]$ is equal to 1 if p is true and 0 otherwise. In words, the empirical risk is the number of misclassified examples in the training data.

The binary perceptron (Rosenblatt, 1957) is an online algorithm that starts from a null model $\boldsymbol{w} = \mathbf{0}$ and iteratively updates its parameters. In Figure 1.2, we present the pseudo-code of this algorithm. At each iteration,

$\boldsymbol{w} \leftarrow \mathbf{0}$
while no convergence
  for each $(\boldsymbol{x}, y) \in \mathcal{D}$
    $\hat{y} \leftarrow F(\boldsymbol{x}; \boldsymbol{w})$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \left(\frac{y-\hat{y}}{2}\right) \cdot \boldsymbol{\Phi}(\boldsymbol{x})$
return $\boldsymbol{w}$

Figure 1.2: Binary perceptron algorithm.

the perceptron draws a training example $(\boldsymbol{x}, y)$ and performs two actions: a prediction $\hat{y}$ is obtained by applying the current model $\boldsymbol{w}$ and, in case of a prediction error, the model is updated towards the misclassified example. Observe that, for positive examples ($y = +1$), the misclassified example features $\boldsymbol{\Phi}(\boldsymbol{x})$ are summed to the model parameters. And, for negative examples ($y = -1$), the features are subtracted from the model parameters. If the predicted output is correct ($\hat{y} = y$), the model $\boldsymbol{w}$ is not updated. This algorithm is proved to converge to a zero-risk solution, if one exists (Novikoff, 1962). Other training algorithms like stochastic gradient descent or support vector machines can also be employed with similar performance guarantees.

Weston and Watkins (1998) propose a generalization of the binary linear discriminant approach for multiclass classification problems. Their approach learns K linear discriminant functions, one for each class. Hence, for each class $k \in \{1, \ldots, K\}$, it learns a parameter vector $\boldsymbol{w}_k$ of a linear discriminant function given by

$$s(\boldsymbol{x}; \boldsymbol{w}_k) = \langle \boldsymbol{w}_k, \boldsymbol{\Phi}(\boldsymbol{x}) \rangle. \tag{1-4}$$

Up to this point, this approach is very similar to the one-vs-all approach cited earlier. However, it differs on the training strategy and also on the prediction function that is given by

$$F(\boldsymbol{x}; \boldsymbol{w}) = \arg\max_{y \in \mathcal{Y}} s(\boldsymbol{x}; \boldsymbol{w}_y), \tag{1-5}$$

where $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K)$ is the concatenation of the parameter vectors for all classes and $\mathcal{Y} = \{1, \ldots, K\}$ is the set of classes. As one can notice, the K discriminant functions are *jointly* employed to perform a prediction.

There are different training algorithms that jointly estimate the discriminant parameters for this multiclass model. Weston and Watkins (1998)

propose an SVM-based formulation by designing joint constraints for each training example, and Crammer and Singer (2001) propose an alternative formulation in order to derive an efficient algorithm for the resulting learning problem. Later, Crammer and Singer (2003) introduce a family of algorithms, called ultraconservatives, for training this kind of joint multiclass model with proven error bounds. In this same work, Crammer and Singer further propose the *margin infused relaxed algorithm* (MIRA), a ultraconservative algorithm that incorporates a generalized notion of margin for multiclass problems. The *multiclass perceptron* is a generalization of the binary perceptron and is a member of the ultraconservative family. We present the pseudo-code of this algorithm in Figure 1.3. As its binary counterpart, the multiclass

$\boldsymbol{w} \leftarrow \mathbf{0}$
while no convergence
  for each $(\boldsymbol{x}, y) \in \mathcal{D}$
    $\hat{y} \leftarrow \arg\max_{y' \in \mathcal{Y}} \langle \boldsymbol{w}_{y'}, \boldsymbol{\Phi}(\boldsymbol{x}) \rangle$
    $\boldsymbol{w}_y \leftarrow \boldsymbol{w}_y + \boldsymbol{\Phi}(\boldsymbol{x})$
    $\boldsymbol{w}_{\hat{y}} \leftarrow \boldsymbol{w}_{\hat{y}} - \boldsymbol{\Phi}(\boldsymbol{x})$
return $\boldsymbol{w}$

Figure 1.3: Multiclass perceptron algorithm.

perceptron is an online algorithm that, on each iteration, picks a training example $(\boldsymbol{x}, y)$ and performs two main steps. First, a prediction $\hat{y}$ is obtained by applying the multiclass prediction function (1-5) using the current joint model $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K)$. Then, if the predicted class $\hat{y}$ is not the correct one y, the joint model $\boldsymbol{w}$ is updated as follows. The correct class model $\boldsymbol{w}_y$ is incremented by the input feature vector $\boldsymbol{\Phi}(\boldsymbol{x})$ and the predicted class model $\boldsymbol{w}_{\hat{y}}$ is decremented by this vector. For all other classes, their models are not updated. This update rule benefits the correct class in detriment of the predicted one in the next predictions regarding the current input features. If the predicted class is correct, the correct and predicted updates cancel each other and actually no update is performed.

Collins (2002b) further generalizes the multiclass perceptron for sequence labeling problems, like POS tagging. Following Weston and Watkins (1998) and Collins (2002b), Altun et al. (2003) propose an SVM-based formulation for sequence labeling and Joachims (2003) develops a related approach for sequence alignment. McDonald et al. (2005) further extend these methods and propose MSTParser, a system that uses a generalization of MIRA to train a dependency parser. Its training algorithm learns a linear discriminant function over head-modifier edges. Given an input sentence $\boldsymbol{x}$ and an edge (i, j) that

corresponds to a candidate dependency between the words x_i and x_j in $\boldsymbol{x}$, they define the following discriminant function

$$s(\boldsymbol{x}, i, j; \boldsymbol{w}) = \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, i, j) \rangle, \tag{1-6}$$

where $\boldsymbol{\Phi}(\boldsymbol{x}, i, j)$ is a feature vector that describes the given dependency edge. The prediction function is then given by

$$F(\boldsymbol{x}; \boldsymbol{w}) = \arg \max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \sum_{(i,j) \in \boldsymbol{y}} s(\boldsymbol{x}, i, j; \boldsymbol{w}), \tag{1-7}$$

where $\mathcal{Y}(\boldsymbol{x})$ is the set of all rooted trees over the words in $\boldsymbol{x}$, i.e., all possible dependency trees for the input sentence. This optimization problem is thus to find a rooted tree such that the sum of its edges scores is maximum. This problem corresponds to the maximum branching problem that is efficiently solved by Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967).

Fernandes and Milidiú (2012) use a generalization of the multiclass perceptron to estimate a parameter vector that minimizes the empirical risk of this DP model. In Figure 1.4, we present the pseudo-code for this algorithm. Again, it is an online algorithm that, at each iteration, performs

$\boldsymbol{w} \leftarrow \mathbf{0}$
while no convergence
  for each $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$
    $\hat{\boldsymbol{y}} \leftarrow \arg\max_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x})} \sum_{(i,j) \in \boldsymbol{y}'} \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, i, j) \rangle$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \sum_{(i,j) \in \boldsymbol{y}} \boldsymbol{\Phi}(\boldsymbol{x}, i, j) - \sum_{(i,j) \in \hat{\boldsymbol{y}}} \boldsymbol{\Phi}(\boldsymbol{x}, i, j)$
return $\boldsymbol{w}$

Figure 1.4: Generalized perceptron algorithm for dependency parsing.

two main actions regarding a training instance $(\boldsymbol{x}, \boldsymbol{y})$: prediction and model update. The predicted tree $\hat{\boldsymbol{y}}$ is obtained by solving (1-7), which is done by Chu-Liu-Edmonds algorithm. The update rule is similar to the one used in the multiclass perceptron. Features present in the edges within $\boldsymbol{y}$ have their weights incremented, and weights for features within $\hat{\boldsymbol{y}}$ are decremented. Edges that are present in both $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are canceled, thus their weights are not updated. Edges not present in either structures are also ignored.

Eventually, it has been realized that these task-specific methods are just instances of a general *structure learning framework*. In this framework, for a given input $\boldsymbol{x}$, the prediction problem is formulated as the following $\boldsymbol{w}$-parameterized optimization problem

$$F(\boldsymbol{x}; \boldsymbol{w}) = \arg \max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} s(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}), \tag{1-8}$$

where $\mathcal{Y}(\boldsymbol{x})$ is the set of possible predictions for the given input $\boldsymbol{x}$; and $s(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w})$ is a $\boldsymbol{w}$-parameterized function that jointly scores an input-output pair $(\boldsymbol{x}, \boldsymbol{y})$. Intuitively, the scoring function measures how well the output structure fits the input. And, it is given by the following linear discriminant function

$$s(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle, \tag{1-9}$$

where $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})$ is an arbitrary joint feature vector representation of the input-output pair. Thus, for a given input, the prediction problem is to find the output with the highest score; where an output score is given by a discriminant function that is linear on some joint feature representation.

The structure learning framework *directly* solves structured problems in a general and principled way. The *structure perceptron* (SPerc) is a general training algorithm that follows the ERM principle to estimate the required parameter vector $\boldsymbol{w}$. This algorithm is presented in Figure 10.4. It is easy

$\boldsymbol{w} \leftarrow \mathbf{0}$
while no convergence
  for each $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$
    $\hat{\boldsymbol{y}} \leftarrow \arg\max_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x})} \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}') \rangle$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}})$
return $\boldsymbol{w}$

Figure 1.5: Structure perceptron algorithm.

to show that the algorithms from Figures 1.2, 1.3 and 1.4 are instances of the SPerc algorithm. Moreover, an extension of Novikoff's theorem (Novikoff, 1962) proves that this algorithm converges to a zero-risk solution, if one exists.

Collins (2002b) proposes the structure perceptron algorithm with an averaging strategy. This is a known strategy used even with the binary perceptron and turns the algorithm more robust. In Figure 1.6, we present the pseudo-code of the averaged SPerc. It is very similar to the algorithm

$\boldsymbol{w}^0 \leftarrow \mathbf{0}$
$t \leftarrow 0$
while no convergence
  for each $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$
    $\hat{\boldsymbol{y}} \leftarrow \arg\max_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x})} \langle \boldsymbol{w}^t, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}') \rangle$
    $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}})$
    $t \leftarrow t + 1$
return $\frac{1}{t} \sum_{k=1}^{t} \boldsymbol{w}^k$

Figure 1.6: Averaged structure perceptron algorithm.

presented in Figure 10.4. Given that the algorithm executes for T iterations, the

averaged SPerc builds a sequence of models $\boldsymbol{w}^0, \ldots, \boldsymbol{w}^T$. Instead of returning the last model $\boldsymbol{w}^T$, like the ordinary SPerc, it returns the average among all built models, that is, $\boldsymbol{w} = \frac{1}{T}\sum_{k=1}^{T} \boldsymbol{w}^k$. Each update in the SPerc algorithm has potentially a big impact on the model parameters. Thus, the averaged algorithm is more robust to noisy examples, and usually performs significantly better than the non-averaged version.

The power of this framework relies mainly on the freedom to design the feature representation and the prediction problem. Specific feature representations allow the design of adherent models for complex SL problems. The decomposition of the joint feature vector along the output structures gives rise to meaningful prediction problems that frequently are reduced to well studied optimization problems. For instance, Collins (2002b) and Altun et al. (2003) model sequence labeling problems through the SL framework, and the resulting prediction problems are solved by the well known Viterbi algorithm; Joachims (2003) develops an approach to learn sequence alignment score functions that are then plugged in the Smith-Waterman alignment algorithm (Smith and Waterman, 1981); and Altun et al. (2007) create a constituent parser whose prediction is performed by the CKY parsing algorithm. This natural fitting between prediction problems and task-meaningful algorithms is no coincidence. It is a consequence of the fact that the SL framework is general yet very flexible.

1.3 Nonlinearity

Linear models are pervasive in ML, mainly because there are efficient training algorithms with proven error bounds to estimate such models. On the other hand, many SL problems are highly non-linear on the available input features. Therefore, when training a linear model within the SL framework, it is necessary to use some feature generation method in order to provide the required nonlinear feature combinations.

Feature generation is frequently solved by a domain expert that generates complex and discriminative feature templates by conjoining input features. Manual template generation is a limited and expensive way to obtain feature templates and is recognized as a modeling bottleneck. Another popular alternative is to employ a kernel function, if the learning algorithm allows it. Besides the fact that kernelized training algorithms are computationally expensive, it is difficult to control the generalization performance of the learned models.

1.4 Entropy-Guided Structure Learning

In this work, we use *Entropy-guided Feature Generation* (EFG), an automatic method to generate feature templates. EFG is based on the conditional entropy of local prediction variables given basic features. It receives a training dataset with basic features and produces a set of feature templates by conjoining features that together are highly discriminative. EFG is based on the same strategy of Entropy-guided Transformation Learning (ETL) (Milidiú et al., 2008; dos Santos and Milidiú, 2009a), which generalizes Transformation-Based Learning (Brill, 1995) by automatically generating rule templates.

We introduce EFG in Fernandes and Milidiú (2012), where we apply it to Portuguese dependency parsing and show that it obtains higher performance than the best available manual templates, when the same basic features are given to both systems. In the Conference on Computational Natural Language Learning (CoNLL) 2012 Shared Task (Pradhan et al., 2012), we propose another EFG-based system that achieves the very first place in this competition (Fernandes et al., 2012b). Here, we experimentally compare EFG to manual template generation and kernel methods on three tasks. In Table 1.1, we summarize these results. Observe that EFG outperforms both alternative

	Alternative System		**EFG**	
Task	**Method**	$\mathbf{F_1}$	$\mathbf{F_1}$	**Error Reduction**
Portuguese DP	Manual Templates	90.06	90.28	2.2%
English Chunking	Kernel	93.48	94.12	9.8%
Portuguese Chunking	Kernel	86.67	87.72	7.9%

Table 1.1: Comparison of EFG with other feature generation methods.

methods on the evaluated datasets. Additionally, it is much cheaper than manual templates and computationally faster than kernel methods.

EFG is easily integrated into the general structure learning framework. We extend this framework by including EFG as a preprocessing step. We denote this extension *Entropy-guided Structure Learning* (ESL) framework. ESL is not restricted to natural language processing tasks. In Figure 1.7, we present the pseudo-code of the extended framework. The training dataset $\mathcal{D}$ is given with the available basic features. Before running the learning algorithm, we apply EFG to generate non-linear features that compose the feature vectors $\mathbf{\Phi}(\mathcal{D}) = \{\mathbf{\Phi}(\boldsymbol{x}, \boldsymbol{y})\}_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}}$ given as input for the learning algorithm.

We evaluate the entropy-guided structure learning framework on nine datasets involving five NLP tasks and four languages. In Table 1.2, we depict the performances achieved by ESL systems compared with the best known

$\boldsymbol{\Phi}(\mathcal{D}) \leftarrow \texttt{EFG}(\mathcal{D})$
$\boldsymbol{w}^0 \leftarrow \mathbf{0}$
$t \leftarrow 0$
while no convergence
  for each $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$
    $\hat{\boldsymbol{y}} \leftarrow \arg\max_{\boldsymbol{y}' \in \mathcal{Y}(\boldsymbol{x})} \langle \boldsymbol{w}^t, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}') \rangle$
    $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}})$
    $t \leftarrow t + 1$
return $\frac{1}{t} \sum_{k=1}^{t} \boldsymbol{w}^k$

Figure 1.7: Entropy-Guided Structure Learning framework.

results for each task. ESL reduces the smallest known error for some tasks and

Task	Language	State of the Art	ESL	
			Accuracy	Error Reduction
POS Tagging	Portuguese	96.94	**97.12**	5.9%
POS Tagging	English	**96.83**	96.72	−3.5%
Text Chunking	Portuguese	87.46	**87.72**	2.1%
Text Chunking	English	**94.21**	94.12	−1.6%
Dependency Parsing	Portuguese	**93.03**	92.66	−5.3%
Quotation Extraction	Portuguese	71.26	**76.80**	19.3%
Coreference Resolution	Arabic	53.55	**54.22**	1.4%
Coreference Resolution	Chinese	62.24	**62.87**	1.7%
Coreference Resolution	English	61.31	**63.37**	5.3%
Coreference Resolution	Multilingual	58.25	**60.15**	4.5%

Table 1.2: Comparison of ESL with state-of-the-art systems.

achieves state-of-the-art comparable results for others. These are remarkable results, specially considering that ESL is relatively simple to be applied on such complex tasks. Moreover, many of the systems that outperform ESL include additional information that could be also included in our systems.

1.5 Contributions

This work has seven main contributions:

1. the ESL framework;
2. a comparison of EFG with manual templates and polynomial kernels;
3. nine ESL based systems for fundamental NLP tasks;
4. state-of-the-art systems for three Portuguese tasks;
5. a novel SL modeling of coreference resolution based on latent trees;

6. ESL-based systems that achieve the best performance on the renowned CoNLL-2012 Shared Task on multilingual coreference resolution;

7. state-of-the-art systems for coreference resolution on Arabic, Chinese and English.

Partial results related to this work have been previously presented in eight important scientific events (Fernandes et al., 2009a,b; dos Santos et al., 2010; Fernandes et al., 2010a,b; Fernandes and Brefeld, 2011; Fernandes and Milidiú, 2012; Fernandes et al., 2012b) and one journal paper (Fernandes et al., 2010c).

Our key contribution is the entropy-guided structure learning framework that extends the general linear discriminative SL framework. ESL employs the entropy-guided feature generation method to solve the problem of non-linear feature generation, a known modeling bottleneck. This framework provides a general machine learning approach that can be applied to supervised structure learning problems.

Our second main contribution is to compare the EFG method with two alternative methods for non-linear feature generation, namely manual templates and polynomial kernel methods. We demonstrate that EFG is superior to both alternatives, since it achieves higher performance. Moreover, it is cheaper than manual templates, faster than kernel methods and avoids the overfitting issue shown by the latter. EFG outperforms the best available manual template set for dependency parsing on the Portuguese dataset provided in the CoNLL-2006 Shared Task. We also compare EFG with polynomial kernel methods for Portuguese and English text chunking. EFG outperforms both methods.

Our third main contribution is the ESL framework instantiation on five NLP tasks and the assessment of the resulting systems by comparing their performances with the best known results on nine datasets involving four languages. These five tasks involve four different structured outputs, namely: sequence, segmentation, rooted tree and clustering. The developed ESL systems present state-of-the-art comparable performances on all evaluated datasets.

Our fourth main contribution is to provide three ESL-based systems that outperform the best previous systems on three Portuguese tasks. On Mac-Morpho, a POS tagging dataset, our system reduces the previous smallest error by 5.9%. On Bosque, a text chunking dataset, the proposed ESL system reduces the previous smallest error by 2.1%. For quotation extraction, our

system reduces the previous smallest error on the GloboQuotes dataset by 19.3%.

Our fifth main contribution is a novel coreference resolution modeling which employs a latent structure in order to control the complexity of the prediction problem. Coreference resolution is a clustering problem and most objective functions for such problems lead to NP-hard optimization problems. In order to design a prediction problem that can be efficiently solved, we represent a coreference cluster as a directed tree that is intuitively adherent to the coreference task.

By employing this latent modeling along with the ESL framework, we achieve the very first place on the renowned CoNLL 2012 Shared Task. The competing systems are ranked by the mean score over three languages: Arabic, Chinese and English. Our ESL systems present an error that is 1.1% smaller than the runner-up competitor on this multilingual task. By further improving our Chinese system, we achieve a 4.5% error reduction over the runner-up system. This relevant achievement constitutes our sixth main contribution.

Our coreference resolution systems achieve the best known performance on the three languages considered in the CoNLL-2012 Shared Task. Our systems reduce the smaller known error by 1.4%, 1.7%, and 5.3%, respectively, on Arabic, Chinese and English. These state-of-the-art systems for multilingual unrestricted coreference resolution comprise our seventh main contribution.

1.6 Dissertation Organization

We use the dependency parsing task as an illustrative application of the main concepts underlying our work. In Chapter 2, we detail the application of the structure learning framework for this task. In this chapter, we also introduce two important extensions to the basic SL framework, namely large margin training and latent structures. In Chapter 3, we describe the proposed entropy-guided feature generation method for structure learning problems. Again, we use DP to illustrate EFG application. We also present in this chapter a comparison of the EFG method to manual templates and kernel methods under the same experimental conditions. In Chapter 4, we detail ESL, the extension of the structure learning framework by incorporating EFG. We also show that the DP system presented in the previous chapters is an instance of this general framework. One key component of ESL is the prediction problem. We discuss some important aspects related to prediction problems in Chapter 5. In this chapter, we also present important examples of prediction problems. Our dependency parsing system is summarized in

Chapter 6. We apply our framework to two part-of-speech tagging datasets. In Chapter 7, we detail these two applications and their experimental results. In Chapter 8, we introduce the application of ESL to two text chunking datasets and the corresponding experimental results. In Chapter 9, we present an ESL-based system to quotation extraction and report on some experiments with a Portuguese dataset. As mentioned before, we achieved the first place in the CoNLL-2012 Shared Task, which was dedicated to multilingual coreference resolution. In Chapter 10, we describe our ESL modeling for this task and the achieved results. This system further extends the ESL framework by introducing latent structures, which are also described. Finally, in Chapter 11, we present our concluding remarks.