PUC

ISSN 0103-9741

Monografias em Ciência da Computação

nº 16/11

Entropy-Guided Feature Generation for Large Margin Structured Learning

Eraldo Luís Rezende Fernandes

Ruy Luiz Milidiú

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 16/11 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena December, 2011

Entropy-Guided Feature Generation for Large Margin Structured Learning *

Eraldo Luís Rezende Fernandes Ruy Luiz Milidiú

efernandes@inf.puc-rio.br , milidiu@inf.puc-rio.br

Abstract. Structured learning consists in learning a mapping from inputs to structured outputs by means of a sample of correct input-output pairs. Many important problems fit in this setting. For instance, dependency parsing involves the recognition of a *tree* underlying a sentence. Feature generation is an important subtask of structured learning modeling. Usually, it is partially solved by a domain expert that builds complex and discriminative feature templates by conjoining the available basic features. This is a limited and expensive way to generate features and is recognized as a modeling bottleneck. In this work, we propose an automatic method to generate feature templates for structured learning algorithms. We denote this method *entropy guided* since it is based on the conditional entropy of local output variables given some basic features. We have evaluated our method on four computational linguistic tasks. We compare the proposed method with two important alternative feature generation methods, namely manual template generation and polynomial kernel functions. Our results show that entropy-guided feature generation outperforms both alternatives and, furthermore, presents additional advantages. The proposed method is cheaper than manual templates and much faster than kernel methods. Furthermore, the developed systems present state-of-the-art comparable performances and, particularly on Portuguese dependency parsing, remarkably reduces the previous smallest error by more than 15%. We further propose to model two complex natural language processing problems that, as far as we know, have never been approached by structured learning methods before. Namely, quotation extraction and coreference resolution.

Keywords: structured learning, feature generation, entropy, natural language processing

Resumo. Aprendizado estruturado consiste em aprender um mapeamento de variáveis de entradas para saídas estruturadas através de exemplos de pares entrada-saída. Vários problemas importantes podem ser modelados desta maneira. Por exemplo, parsing de dependência envolve o reconhecimento de uma *árvore* a partir de uma frase. Geração de atributos é uma sub-tarefa importante da modelagem em aprendizado estruturado. Geralmente, esta sub-tarefa é realizada por um especialista que constrói gabaritos de

* This work was partially funded by grants from CNPq (557.128/2009-9) and FAPERJ (E-26/170028/2008). The first author was suported by a CNPq doctoral fellowship, a CAPES doctoral internship grant, and by Instituto Federal de Educação, Ciência e Tecnologia de Goiás.

atributos complexos e discriminativos através da combinação dos atributos básicos disponíveis. Esta é uma forma limitada e cara para geração de atributos e é reconhecida como um gargalo de modelagem. Neste trabalho, propomos um método automático para geração de atributos para problemas de aprendizado estruturado. Denotamos este método como *guiado por entropia* já que ele é baseado na entropia condicional de variáveis locais de saída dados alguns atributos. Nós avaliamos nosso método em quatro tarefas de linguística computacional. Comparamos o método proposto com dois métodos alternativos para geração de atributos: geração manual e funções de kernel polinomial. Estes resultados mostram que o método de geração de atributos guiado por entropia é melhor do que os dois métodos alternativos e ainda apresenta vantagens adicionais. O método proposto é mais barato do que o método manual e mais rápido que o método baseado em kernel. Além disso, os sistemas desenvolvidos apresentam resultados comparáveis com os melhores sistemas atualmente e, particularmente para parsing de dependência para Português, obtivemos uma redução de erro de mais de 15% comparado com o melhor resultado até o momento. Ainda propomos a modelagem de duas tarefas de processamento de linguagem natural que, até onde podemos dizer, nunca foram modeladas através de métodos de aprendizado estruturado. Estas tarefas são extração de citações e resolução de coreferência.

Palavras-chave: aprendizado estruturado, geração de atributos, entropia, processamento de linguagem natural

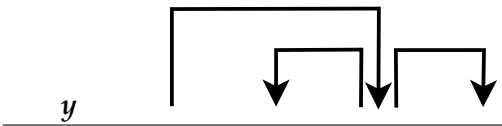
In charge of publications :

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Contents

1 Introduction

In the last decades, machine learning has been successfully applied on many fields from natural language processing and information extraction to computer vision and computational biology. Many important machine learning problems involve the prediction of complex structures that comprise interdependent variables. Dependency parsing (DP), for instance, is to derive a syntactic structure that identifies the syntactic *head* of each token in a given input sentence. This structure is called *dependency tree* and is a rooted tree whose nodes are the sentence tokens. Let $\boldsymbol{x} = (x_0, x_1, \ldots, x_T)$ be an input sentence, where x_i is the i-th token and x_0 is a dummy token which is always the root of the dependency tree. We define $\mathcal{Y}(\boldsymbol{x})$ as the set of all rooted trees whose nodes are tokens in the sentence $\boldsymbol{x}$ and the root node is x_0. For any dependency tree $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$, we say that $(i, j) \in \boldsymbol{y}$ whenever token x_i is the head of token x_j in the tree $\boldsymbol{y}$. Figure 1 shows the sentence "*John saw Mary*", the dependency tree $\boldsymbol{y} = \{(0,2), (2,1), (2,3)\}$, and the corresponding token heads.



$\boldsymbol{i}$	0	1	2	3
$\boldsymbol{x}$	*root*	John	saw	Mary
Head	–	2	0	2

Figure 1: Dependency tree of a sentence

Most of the best performing systems for such structured problems are complex machine learning systems that combine several binary classifiers. Additionally, in order to consider the natural interdependencies among output variables, the binary classifiers are trained by task specific strategies that share information or enforce constraints among the basic classifiers. For instance, [1] propose a DP system 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 and previous edge predictions are used to consider some interdependency between token heads.

In the last few years, machine learning methods that *directly* solve structured problems have emerged. [2] proposed an SVM formulation for multiclass problems that *jointly* trains some linear functions, one for each class. Then, these functions are also jointly used to perform predictions. Before that, such problems had been solved by independently trained binary classifiers. Additionally, some voting scheme enforces the constraint that only one class has to be chosen for a given input. [2] showed that their approach outperforms this ensemble one. Next, [3] proposed the structured perceptron algorithm for sequence labeling problems. This algorithm is a generalization of the binary perceptron and learns the parameters of a linear discriminant function that, given an input sequence, discriminates the correct tag sequence from the alternative ones by means of a dynamic programming algorithm. Following [2] and [3], [4] proposed an SVM-based formulation for sequence labeling and [5] developed a related approach for sequence alignment.

In the same line of these previous methods, [6] proposed MSTParser, an online algorithm to train a dependency parser that recast the parsing problem as a maximum branching problem [7]. They propose to learn a scoring function $s(i, j)$ over all candidate head-dependent edges (i, j) such that the prediction problem $F(\boldsymbol{x})$, that is to predict a dependency tree for an input sentence $\boldsymbol{x}$, is to find the maximum-score tree among the

valid rooted trees:

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

where $s(\boldsymbol{y}) = \sum_{(i,j)\in\boldsymbol{y}} s(i,j)$ is the score of a candidate tree $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$, i.e., the sum of its edges scores. This is the well studied maximum branching problem that can be efficiently solved by Chu-Liu-Edmonds' algorithm [8,9]. There is also an improved implementation of this algorithm by [7].

Obviously, the scoring function is key for MSTParser and has to generalize over any possible input sentence. For a sentence $\boldsymbol{x}$ and a candidate edge (i,j), MSTParser determines $s(i,j)$ by means of M real-valued feature functions (or, simply, *features*) denoted by $\phi_m(\boldsymbol{x},i,j)$, for $m = 1,\ldots,M$. These features describe the candidate edge from token x_i to token x_j in a meaningful and general way. MSTParser uses several features like, for instance, the i-th word, the j-th word, the part-of-speech of the i-th word, the distance from x_i to x_j, the relative order between x_i and x_j, among others. Then, the scoring function is defined as a weighted sum of the edge features:

$$s(i,j) = \sum_{m=1}^{M} w_m \cdot \phi_m(\boldsymbol{x},i,j),$$

where $\boldsymbol{w} = (w_1,\ldots,w_M)$ comprises the model parameters that have to be learned. In that way, the learning problem is to determine a parameter vector $\boldsymbol{w}$ such that the discriminant $F(\cdot)$ has small empirical risk, which is given by $\mathcal{R}(\mathcal{D},\boldsymbol{w}) = \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \mathbf{1}_{[\boldsymbol{y}\neq F(\boldsymbol{x})]}$, where $\mathcal{D} = \{(\boldsymbol{x},\boldsymbol{y})\}$ is the training dataset with correct sentence-tree pairs; and $\mathbf{1}_{[p]}$ is 1 if p is true and 0 otherwise.

Eventually, researchers have realized that these mentioned task-specific structured learning methods were highly related and could be unified into a general framework. This generalized framework for structured machine learning is based on a prediction function that is formulated as a discriminant optimization problem. The objective function of this optimization problem is a linear function on a joint feature vector $\boldsymbol{\Phi}(\boldsymbol{x},\boldsymbol{y}) = (\phi_1(\boldsymbol{x},\boldsymbol{y}),\ldots,\phi_M(\boldsymbol{x},\boldsymbol{y}))$, for given input $\boldsymbol{x}$ and output structure $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$. Thus, for an input $\boldsymbol{x}$, the *discriminant problem* is defined as:

$$F(\boldsymbol{x}) = \arg\max_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})} \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x},\boldsymbol{y})\rangle, \tag{2}$$

where $\mathcal{Y}(\boldsymbol{x})$ is the set of all feasible output structures for the input $\boldsymbol{x}$ and $\langle\cdot,\cdot\rangle$ is the scalar product operator. Regarding dependency parsing, we can rewrite (1) in the form of (2) by letting $\phi_m(\boldsymbol{x},\boldsymbol{y}) = \sum_{(i,j)\in\boldsymbol{y}} \phi_m(\boldsymbol{x},i,j)$ be the *global* feature function that combines the feature m on all edges into a unique value. In that way, we have that

$$s(\boldsymbol{y}) = \sum_{(i,j)\in\boldsymbol{y}} \sum_{m=1}^{M} w_m \cdot \phi_m(\boldsymbol{x},i,j) = \sum_{m=1}^{M} w_m \sum_{(i,j)\in\boldsymbol{y}} \phi_m(\boldsymbol{x},i,j) = \sum_{m=1}^{M} w_m \cdot \phi_m(\boldsymbol{x},\boldsymbol{y}),$$

which is equal to $\langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x},\boldsymbol{y})\rangle$.

The power of this framework relies mainly on the freedom to define the feature mapping $\boldsymbol{\Phi}(\boldsymbol{x},\boldsymbol{y})$, the output space $\mathcal{Y}(\boldsymbol{x})$, and how the features are decomposed along the output structures $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$. In MSTParser, for instance, the output space comprises rooted trees and the features are decomposed along the candidate edges. Thus, the discriminant problem is reduced to the maximum branching problem that can be efficiently solved. Task modeling through this framework involves a trade-off between feature meaningfulness and feature simplicity. Features shall be representative enough w.r.t. the task at

hand, but, conversely, they shall also be local or well behaved enough w.r.t. the output space, to avoid intractable discriminant problems.

There are some training algorithms that learn the parameter vector $\boldsymbol{w}$ from a given training dataset $\mathcal{D} = \{(\boldsymbol{x}, \boldsymbol{y})\}$ of correct input-output pairs. For instance, [3] proposed the structured perceptron algorithm, a generalization of the well known binary perceptron algorithm for sequence labeling problems. The structured perceptron can be easily applied to any structured problem. [3] also proved that the structured perceptron converges to a zero-error solution, if one exists. [10] proposed the margin infused relaxed algorithm (MIRA), an online algorithm to train structured models for multiclass problems. MIRA can be extended for virtually any structured problem and, for instance, is the MSTParser's training algorithm. [10] also proved some mistake bounds for an algorithm class called *ultraconservative*. This class includes MIRA and structured perceptrons. [11] formulated the structured learning problem through a regularized maximum-margin framework, inspired on the binary SVM formulation. They also proposed a cutting plane method to efficiently solve this problem. However, this method still requires more computational power and memory than online algorithms like structured perceptron and MIRA. Additionally, the online algorithms are simpler to implement than the SVM-based one.

Feature generation is an important subtask of structured learning modeling. Usually, it is partially solved by a domain expert that generates complex and discriminative feature templates by conjoining the available basic features. This is a limited and expensive way to obtain feature templates and is recognized as a modeling bottleneck. A common alternative is to employ a kernel function, when the learning algorithm allows so. However, one drawback of kernels is related to overtraining since it is difficult to finely tune the trade-off between representation power and unseen data generalization. We propose an automatic method to generate feature templates for structured learning algorithms. The method receives as input the training dataset with basic features and produces a set of feature templates by conjoining basic features that are highly discriminative together. We denote this method *entropy-guided* feature generation (EFG) since it is based on the conditional entropy of local output variables given some basic features. Preliminary results on some natural language processing (NLP) problems indicate that EFG outperforms manually created templates and kernel-based methods. We report on experiments on four datasets involving three NLP tasks and two languages.

We further propose to model two complex natural language processing problems that, as far as we know, have never been approached by structured learning methods before. Namely, quotation extraction and coreference resolution. We plan to model different discriminant problems for these two problems and also for more basic problems. For instance, text chunking and named entity recognition are two tasks that have been recurrently recast as sequence tagging problems, even not being so. The aim, on both tasks, is to identify chunks of words that have a specific meaning in the sentence. Another proposed contribution of this work consists in modeling these two tasks through the well known weighted interval scheduling problem. In this way, we will be able to use more powerful features.

In this work, we make use of a generalization of Collins' structured perceptron with large margin strategy in order to train our models. In Section 2, we describe this algorithm and its kernelized version that is important to understand the proposed entropy-guided feature generation approach. Two main alternative approaches to feature generation are presented in Section 3, namely manual feature generation and kernel methods. In this section, we also discuss previous work based on these alternative approaches. In Section 4, we present the proposed entropy-guided feature generation method. The pro-

posed contributions related to task modeling are described in Section 5. Finally, preliminary results are discussed in Section 6 and concluding remarks are presented in Section 7.

2 Structured Perceptron

The structured perceptron algorithm [3] is analogous to its binary counterpart. Given a training sample $\mathcal{D}$ of correct input-output pairs, the algorithm generates a sequence $\boldsymbol{w}^0 = \mathbf{0}, \boldsymbol{w}^1, \ldots, \boldsymbol{w}^K$ of models. At each iteration $k = 0, \ldots, K$, a training instance $(\boldsymbol{x}, \boldsymbol{y})$ is drawn from $\mathcal{D}$ and two major steps are performed, that is:

(1) *Prediction*: $\hat{\boldsymbol{y}} \leftarrow F(\boldsymbol{x})$, using the current model $\boldsymbol{w}^k$,

(2) *Model Update*: $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k + \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}})$.

Note that, when a correct prediction is made, that is $\hat{\boldsymbol{y}} = \boldsymbol{y}$, the model does not change, that is $\boldsymbol{w}^{k+1} \leftarrow \boldsymbol{w}^k$. When the prediction is wrong, the update rule favors the correct output $\boldsymbol{y}$ over the predicted one $\hat{\boldsymbol{y}}$. Regarding binary features, for instance, the update rule increases the weights of features that are present in $\boldsymbol{y}$ but missing in $\hat{\boldsymbol{y}}$ and decreases the weights of features that are present in $\hat{\boldsymbol{y}}$ but not in $\boldsymbol{y}$. The weights of features that are present in both $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are not changed. A simple extension of Novikoff's theorem [12] shows that the structured perceptron is guaranteed to converge to a zero loss solution, if one exists, in a finite number of steps [4, 13]. [10] further prove some mistake bounds for the structured perceptron algorithm.

2.1 Large Margin Training

The structured perceptron algorithm finds a classifier with no concern about its margin. However, it is well known that large margin classifiers provide better generalization performance on unseen data. MIRA [10], which is MSTParser's training algorithm, is an generalization of the structured perceptron algorithm that generates a large margin classifier. However, in this work, we use another large-margin generalization of the structured perceptron that is based on the margin rescaling technique for structured support vector machines [14]. For a training instance $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$, instead of the ordinary discriminant problem $F(\boldsymbol{x})$, we use the following *loss-augmented* discriminant problem in the step (1) of the structured perceptron algorithm:

$$F_\ell(\boldsymbol{x}) = \arg\max_{\tilde{\boldsymbol{y}} \in \mathcal{Y}(\boldsymbol{x})} \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, \tilde{\boldsymbol{y}}) \rangle + \ell(\boldsymbol{y}, \tilde{\boldsymbol{y}}),$$

where $\ell(\cdot, \cdot) \geq 0$ is a given loss function that measures the difference between a candidate tree and the correct one. For instance, the most common loss function for dependency trees just counts how many tokens have been assigned for an incorrect head, that is $\ell(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \sum_{i=1}^{T} \mathbf{1}_{[y(i) \neq \tilde{y}(i)]}$, where $y(i)$ is the head of the i-th token in $\boldsymbol{y}$.

By using the loss-augmented discriminant, a training instance $(\boldsymbol{x}, \boldsymbol{y})$ implies a model update whenever the current model does not respect the following margin constraint:

$$s(\boldsymbol{y}) - s(\tilde{\boldsymbol{y}}) \geq \ell(\boldsymbol{y}, \tilde{\boldsymbol{y}}), \ \forall \tilde{\boldsymbol{y}} \in \mathcal{Y}(\boldsymbol{x}).$$

If a model respects this constraint then the current discriminant function $F(\boldsymbol{x})$ separates the correct output $\boldsymbol{y}$ from every alternative $\tilde{\boldsymbol{y}} \in \mathcal{Y}(\boldsymbol{x})$ by a margin as large as the loss

$\ell(\boldsymbol{y}, \tilde{\boldsymbol{y}})$ between them. In that way, the training algorithm incorporates *some* information about the *structured* empirical risk $\mathcal{R}(\mathcal{D}, \boldsymbol{w})$ of the current model, defined as

$$\mathcal{R}(\mathcal{D}, \boldsymbol{w}) = \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \ell(\boldsymbol{y}, F(\boldsymbol{x})).$$

2.2 Kernelization

Analogously to the binary perceptron algorithm, its structured generalization can easily be kernelized. Given the sequence $(\boldsymbol{x}^1, \boldsymbol{y}^1, \hat{\boldsymbol{y}}^1), \ldots, (\boldsymbol{x}^k, \boldsymbol{y}^k, \hat{\boldsymbol{y}}^k)$ of inputs, correct outputs and predicted outputs considered by the training algorithm up to time k, the parameter vector at this point can be defined as

$$\boldsymbol{w}^k = \mathbf{0} + \sum_{j=1}^{k-1} \left[\boldsymbol{\Phi}(\boldsymbol{x}^j, \boldsymbol{y}^j) - \boldsymbol{\Phi}(\boldsymbol{x}^j, \hat{\boldsymbol{y}}^j) \right].$$

The algorithm can keep track of how many times each alternative output $\hat{\boldsymbol{y}}$ has been predicted instead of the correct output $\boldsymbol{y}$ for each example pair $(\boldsymbol{x}, \boldsymbol{y})$ by means of counters $\alpha_{\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{y}}}$. Thus, the parameter vector can be rewritten as

$$\boldsymbol{w} = \sum_{\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{y}}} \alpha_{\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{y}}} \cdot \left[\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}) \right], \tag{3}$$

which is called the *dual model representation*. For many structured problems, the output space $\mathcal{Y}(\boldsymbol{x})$ is huge – exponential on the input size – or even infinity. Thus, the dual model representation may comprise an intractable number of parameters. However, these parameters are initially *zero* for all $\boldsymbol{x}, \boldsymbol{y}, \hat{\boldsymbol{y}}$ and only need to be instantiated once the respective triplet is actually seen.

Using (3), the objective function of the discriminant problem can also be rewritten as

$$\langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}', \boldsymbol{y}') \rangle = \sum_{\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{y}}} \alpha_{\boldsymbol{x},\boldsymbol{y},\hat{\boldsymbol{y}}} \cdot \left[\langle \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{\Phi}(\boldsymbol{x}', \boldsymbol{y}') \rangle - \langle \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}), \boldsymbol{\Phi}(\boldsymbol{x}', \boldsymbol{y}') \rangle \right],$$

which depends only on inner products of feature vectors of the form $\langle \boldsymbol{\Phi}^a, \boldsymbol{\Phi}^b \rangle$, where $\boldsymbol{\Phi}^a$ and $\boldsymbol{\Phi}^b$ are shortcuts to, respectively, $\boldsymbol{\Phi}(\boldsymbol{x}^a, \boldsymbol{y}^a)$ and $\boldsymbol{\Phi}(\boldsymbol{x}^b, \boldsymbol{y}^b)$ for any input-output pairs $(\boldsymbol{x}^a, \boldsymbol{y}^a)$ and $(\boldsymbol{x}^b, \boldsymbol{y}^b)$. Following the *kernel trick* [15], the inner products of feature vectors can then be replaced by appropriate kernel functions

$$K(\boldsymbol{\Phi}^a, \boldsymbol{\Phi}^b) = \langle \boldsymbol{\Psi}(\boldsymbol{\Phi}^a), \boldsymbol{\Psi}(\boldsymbol{\Phi}^b) \rangle,$$

where $\boldsymbol{\Psi}(\cdot)$ expands elements from the original feature vector space $\boldsymbol{\Phi}(\cdot, \cdot)$ to a much higher dimensional space. The kernel trick relies on the kernel function $K(\cdot, \cdot)$ to efficiently compute inner products in the high dimensional space of $\boldsymbol{\Psi}$ with no need to explicitly expand the original space of $\boldsymbol{\Phi}$.

The most successful kernel type for NLP problems is the *polynomial kernel*. Regarding binary features, a polynomial kernel of degree d expands the original feature space by conjoining original features through all possible combinations with up to d features. The polynomial kernel of degree d can be efficiently computed by

$$K_d(\boldsymbol{\Phi}^a, \boldsymbol{\Phi}^b) = \left(\langle \boldsymbol{\Phi}^a, \boldsymbol{\Phi}^b \rangle + 1 \right)^d,$$

which involves only an inner product in the original feature space, a sum of a constant, and an exponentiation. For instance, if $d = 2$ and the original space has exactly 3 *binary*

features, then the *explicit* polynomial kernel expansion of $\mathbf{\Phi}(\boldsymbol{x}, \boldsymbol{y}) = (\phi_1, \phi_2, \phi_3)$ corresponds to $\mathbf{\Upsilon}(\mathbf{\Phi}(\boldsymbol{x}, \boldsymbol{y})) = (1,\ \phi_1,\ \phi_2,\ \phi_3,\ \phi_1\phi_2,\ \phi_1\phi_3,\ \phi_2\phi_3)$.

In general, just like features, kernel functions are also decomposed along the output structures. Thus, the dual model representation can be even more sparse by using α counters for each individual element that appears in $(\boldsymbol{x}, \boldsymbol{y})$ but not in $(\boldsymbol{x}, \hat{\boldsymbol{y}})$, or vice-versa. This decomposition is task dependent, and, e.g., the individual elements of dependency trees are edges.

3 Related Work

In this section, we discuss previously proposed approaches to perform feature generation for structured problems. We focus on two main approaches: manually generated templates and polynomial kernel functions.

Transformation based learning (TBL) was proposed by [16] for part-of-speech tagging. TBL is a machine learning algorithm tailored for sequence labeling problems and has been applied to several NLP problems. For instance, part-of-speech tagging [16], text chunking [17], and named entity recognition [18]. TBL learns a sequence of transformation rules that iteratively correct errors in the training corpus by means of a set of rule *templates* given as input. These templates are manually created by conjoining basic features to form a discriminative, compound feature. Entropy guided transformation learning (ETL) [19] generalizes TBL by automatically generating the required templates. ETL employs the information gain measure, which is based on entropy, in order to select feature combinations. In this work, we apply this strategy to structured learning algorithms.

TBL is never applied without feature templates. Conversely, many other machine learning algorithms, like SVM and Winnow for instance, do not require feature templates and can be directly applied on basic features. However, many systems based on these algorithms also employ templates to improve their performances. Some of the best performing systems for text chunking [20, 21], dependency parsing [22], among several others, make use of simple, manually created feature templates. This is a limited and expensive way to obtain feature templates and is recognized as a modeling bottleneck.

One alternative to manually created templates is the use of polynomial kernel functions. Kernel functions allow the efficient computation of all possible feature combinations with a given maximum number of features d. Many NLP tasks have been successfully approached by kernel methods with polynomial kernel functions. Just to name a few, [20] for text chunking, [23] for named entity recognition, and [1] for dependency parsing are all SVM-based systems with 2nd-degree polynomial kernel that achieve state-of-the-art performances. However, a key issue with polynomial kernel functions is the difficult to finely tune the trade-off between model representation power and overtraining. The user has no control on which feature combinations are used, i.e., the kernel function always computes *all* possible feature combinations. The greater the degree of the kernel, the greater its representation power but also its overtraining sensitivity.

4 Entropy-Guided Feature Generation

Feature generation is an important subtask of structured learning modeling. Usually, a task dataset includes some basic features that are either naturally included in the very

phenomenon of interest, like words for NLP tasks; simply derived from other basic features, like capitalization information; or automatically generated by external systems, like part-of-speech tags. However, using basic features independently is not enough to achieve state-of-the-art results. Thus, it is necessary to conjoin basic features to improve performance. Frequently, a domain expert manually generates feature templates by conjoining the given basic features. MSTParser, for instance, uses 21 feature templates that were manually created from basic features given in the input dataset. In this section, we describe the proposed entropy-guided feature generation method for structured learning, that automatically create feature templates by conjoining basic features that are highly discriminative together. The method uses decision tree induction to incorporate the conditional entropy of local decision variables given input features.

Decision tree (DT) learning is one of the most widely used machine learning algorithms. It performs a partitioning of the training set using principles of information theory. The learning algorithm executes a general to specific search of a feature space. The most informative feature is added to a tree structure at each step of the search. Information gain, which is based on the data entropy, is normally used as the informativeness measure. The objective is to construct a tree, using a minimal set of features, that efficiently partitions the training set into classes of observations. Usually, after the tree is grown, a pruning step is carried out in order to avoid overfitting.

Information gain is based on entropy and is a key strategy for feature selection. The most popular decision tree learning algorithms [24, 25] use this measure. Hence, they provide a quick way to obtain entropy guided feature selection. We propose a new automatic feature generation method for structured learning algorithms. The key idea is to use decision tree induction to obtain new features by conjoining the given basic features. One of the most used algorithms for DT induction is C4.5 [24]. We use Quinlan's C4.5 system to obtain the required entropy guided selected features. The same strategy has been used for automatic template generation in entropy guided transformation learning [19] that generalizes transformation based learning [16].

The first step of our method is to train a decision tree on a dataset where each example comprises the basic features of one *local decision variable* of the discriminant problem. Regarding dependency parsing, for instance, the local decision variables correspond to edges, since the discriminant problem is to choose some edges from all possible edges between tokens. Thus, for each sentence in the input dataset, we generate an example for each pair of tokens. Local decision variables coincide with the decomposition of the feature vectors along the output structure. These local variables will be predicted by the DT algorithm and are also the individual decision variables in the discriminant algorithm. In dependency parsing, for instance, there is a decision variable for each candidate edge, that is for each example, and it is a binary variable that indicates whether an edge is included in the solution or not. Hence, the DT algorithm will learn a decision tree to predict whether an edge is valid or not.

From the learned DT, our method uses a very simple tree decomposition scheme to extract templates. The decomposition process is based on a depth-first traversal of the DT and thus can be recursively described as in the following. For each visited node, a new template is created by conjoining the node feature with its parent template. To limit the maximum template length, we use pruned trees. Figure 2 illustrates our method applied to the DP task. The tree in the left side of the figure uses four basic features (rectangular nodes): `dist` is the absolute distance between the head and the dependent token, `side` is the side of the head token relative to the dependent token, `dep-pos` and `head-pos` are the part-of-speech tag of the dependent and the head token, respectively. The round nodes are the decision variable values. The generated templates are listed in

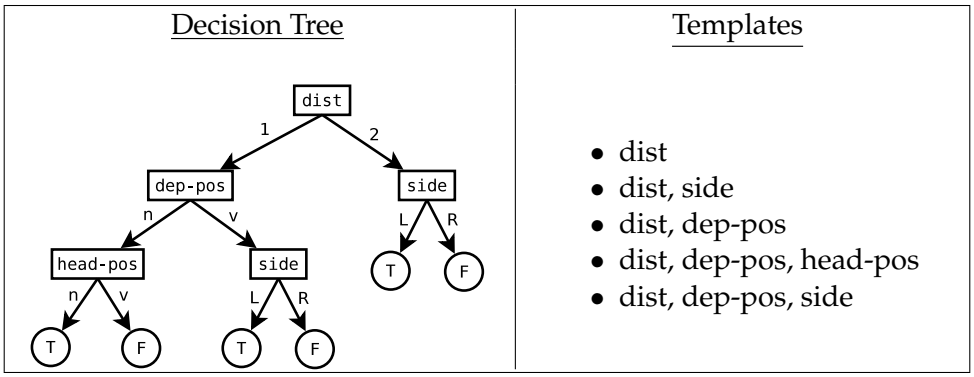


Figure 2: Feature generation from a decision tree

the right side of the figure. In other words, we create a template for each path from the root to every other DT node, ignoring the feature values at the DT edges, thus using only the node features.

5 Task Modeling

In this work, we propose machine learning systems based on structured perceptron for several NLP problems. In this section, we describe the structured modeling for these tasks. Some modeling strategies are applied to more than one task, thus we describe only the general modeling approach and, later in Section 6, we describe the specific details for each task.

5.1 Sequence Labeling

Sequence labeling learning [26] is to find a mapping from an input sequence $\boldsymbol{x} = (x_1, \ldots, x_T)$ to a sequence of labels $\boldsymbol{y} = (y_1, \ldots, y_T)$, where $y_i \in \Sigma$. That is, each element of $\boldsymbol{x}$ is annotated with an element of the output alphabet Σ which denotes the set of labels. As before, we denote the set of all possible labellings for $\boldsymbol{x}$ by $\mathcal{Y}(\boldsymbol{x})$.

In order to model the sequence labeling task through the structured learning framework, we need to define the feature vector $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})$ and its decomposition along the sequence $\boldsymbol{y}$, so that the discriminant problem $F(\boldsymbol{x}) = \arg\max_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle$ can be efficiently solved. We use the decomposition scheme from [3], which relies on a Markov-like property. The best scoring label of any token is determined by parameters that depend only on the *constant* input sequence $\boldsymbol{x}$, the token label itself and the previous token label. In that way, the discriminant problem can be solved by a dynamic programming algorithm that is equivalent to Viterbi decoding for hidden Markov models.

In that way, the feature vector $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})$ comprises two feature subsets: $\boldsymbol{\Phi}^{\text{obs}}(\boldsymbol{x}, \boldsymbol{y})$ for observation features that depend only on the input $\boldsymbol{x}$ and individual token labels; and $\boldsymbol{\Phi}^{\text{trans}}(\boldsymbol{y})$ for transition features that depend on two consecutive labels. For each possible pair of labels $\sigma, \tau \in \Sigma$, the transition feature $\phi^{\text{trans}}_{\sigma\tau}(\boldsymbol{y})$ indicates how many label pairs within $\boldsymbol{y}$ are consecutively labeled σ and τ. So, this feature is decomposed along $\boldsymbol{y}$ as

$$\phi^{\text{trans}}_{\sigma\tau}(\boldsymbol{y}) = \sum_{i=1}^{T} \left(\mathbf{1}_{[y_{i-1}=\sigma]} \cdot \mathbf{1}_{[y_i=\tau]} \right).$$

Thus, we have that $\boldsymbol{\Phi}^{\text{trans}}(\boldsymbol{y}) = (\phi^{\text{trans}}_{\sigma\tau}(\boldsymbol{y}))_{\sigma,\tau \in \Sigma}$.

The observation features are defined by means of M *binary* features. Each token comprises a subset of such features. For instance, we can have the following binary features: "*the token word is light*", "*the token part-of-speech is verb*", "*the word of the previous token is the*", and so on. Hence, we usually have millions of such features in the training corpus, but just a small fraction of them occur in each token. We represent a binary feature by its index $m \in \{1, \ldots, M\}$. Each binary feature corresponds to $|\Sigma|$ observation features, one for each state $\sigma \in \Sigma$. The observation feature denoted by $\phi^{\text{obs}}_{\sigma,m}(\boldsymbol{x}, \boldsymbol{y})$ determines how many times within $(\boldsymbol{x}, \boldsymbol{y})$ the feature m occurs in a token labeled as σ. Therefore, by letting $\pi(x_i)$ be the set of binary features in the i-th token of $\boldsymbol{x}$, we can express the decomposition of an observation feature along $\boldsymbol{y}$ by

$$\phi^{\text{obs}}_{\sigma,m}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{T} \left(\mathbf{1}_{[y_i = \sigma]} \cdot \mathbf{1}_{[m \in \pi(x_i)]} \right).$$

We then define the observation feature vector as $\boldsymbol{\Phi}^{\text{obs}}(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{\Phi}^{\text{obs}}_{\sigma}(\boldsymbol{x}, \boldsymbol{y}))_{\sigma \in \Sigma}$, where $\boldsymbol{\Phi}^{\text{obs}}_{\sigma}(\boldsymbol{x}, \boldsymbol{y}) = (\phi^{\text{obs}}_{\sigma,1}(\boldsymbol{x}, \boldsymbol{y}), \ldots, \phi^{\text{obs}}_{\sigma,M}(\boldsymbol{x}, \boldsymbol{y}))$. And, finally, the complete feature vector is $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) = \big(\boldsymbol{\Phi}^{\text{obs}}(\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{\Phi}^{\text{trans}}(\boldsymbol{x}, \boldsymbol{y})\big)$.

For sequence labeling, we use the loss function $\ell(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \sum_{i=1}^{T} \mathbf{1}_{[y_i \neq \tilde{\boldsymbol{y}}_i]}$ that counts the number of mislabeled tokens.

5.2 Fixed Nodes Tree

Dependency parsing consists in predicting a rooted tree underlying a given sentence. The nodes of this tree are always fixed, they are the tokens within the sentence. Thus, we name this type of structure *fixed nodes tree*, in contrast to other structure, which we discuss later, whose nodes must be determined by the discriminant problem. Since we have already presented this modeling approach in the context of DP, in this section, we just summarize it using more general designations.

Let $\boldsymbol{x} = (x_0, x_1, \ldots, x_T)$ be an input sequence, where x_i is the i-th token and x_0 is a dummy token which is always the root of the tree. We define $\mathcal{Y}(\boldsymbol{x})$ as the set of all rooted trees whose nodes are tokens in $\boldsymbol{x}$ and the root node is x_0. A tree $\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})$ is represented as a set of directed edges (i, j), where i and j are token *indexes* in $\boldsymbol{x}$. Each edge (i, j) is defined by M real-valued features denoted by $\phi_m(\boldsymbol{x}, i, j)$, for $m = 1, \ldots, M$. These *local* features are combined into *global* features by letting

$$\phi_m(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(i,j) \in \boldsymbol{y}} \phi_m(\boldsymbol{x}, i, j).$$

Therefore, the complete joint feature vector is given by $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) = (\phi_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, \phi_M(\boldsymbol{x}, \boldsymbol{y}))$. The loss function is given by $\ell(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \sum_{j=1}^{T} \mathbf{1}_{[y(j) \neq \tilde{\boldsymbol{y}}(j)]}$, where $y(j)$ is the unique token i such that $(i, j) \in \boldsymbol{y}$.

5.3 Sequence Segmentation

Text chunking and named entity recognition are some examples of tasks that comprise the identification of *segments* within a sentence. Named entity recognition, for instance, is to identify mentions of real-world entities and to classify them on categories like person, location, and company. Text chunking consists in identifying *chunks* of tokens that present strong syntactic relation and classifying them according to its syntactic function in the sentence.

Usually, segmentation tasks like those are recast as sequence labeling problems by means of the so called IOB tagging style. The IOB tagging style specifies three types of tags: B for tokens that start a segment; I for tokens that continue a segment; and O for tokens that are not part of any segment. Furthermore, the IOB2 tagging style extends IOB by attaching a type to B and I tags, so that one can segment a sequence and simultaneously classify the segments.

These tagging styles have been proposed to allow the application of ordinary classification approaches to sequence segmentation tasks. However, it is clear that segmentation is quite different from labeling. Thus, in this work, we propose to solve segmentation tasks through structured learning in a more natural way. We can model this problem by somehow generating segment candidates and reducing the discriminant problem to the weighted interval scheduling problem (WIS). If it is not possible to filter which segments are valid candidates, one can generate all $|T|^2$ possible candidates. Since, for many problems, sequences are short and WIS can be solved in linear time for ordered segments, this approach is perfectly feasible.

Quotation extraction consists in identifying quotes and their authors within a given document. Quotes cannot overlap among them and each quote must be associated to exactly one author. We also propose to model quotation extraction through sequence segmentation. Again, we can generate candidates for quotes by means of heuristics or simply by generating all possible segments. Furthermore, for each pair of candidate quote and author (authors are given), we generate a candidate quotation whose segment corresponds to the span of the quote in the document. In that way, the WIS algorithm will choose at most one quotation among all candidates for one specific quote. Moreover, the discriminant algorithm will never choose overlapping quotes, since this is a constraint in the algorithm.

5.4 Variable Nodes Tree

Some NLP problems consist in identifying a tree structure underlying a sentence such that tokens are tree leaves and the internal nodes need be determined. This type of structure is substantially different from dependency trees. Full parsing and clause identification are some examples of such tasks. Usually, there are specific algorithms for each task. For clause identification, for instance, a dynamic programming algorithm has been proposed by [21]. We propose to apply the EFG approach to clause identification for the English and Portuguese languages.

5.5 Clustering

Coreference resolution [27] consists in, given a set of entity mentions within a document, identifying which subsets of mentions are related to the same real-world entity. This problem can be naturally modeled as clustering of mentions. However, the number of clusters is unknown and the optimization problem is NP-hard. On the other hand, several authors [28–31] have reported that structured learning is feasible even with approximate discriminants. These studies have shown that the learning algorithm is able to adjust the feature parameters so that an approximate algorithm can still correctly solve the discriminant problem. This occurs because the feature parameters are adjusted according with the mistakes of the discriminant algorithm. Thus, we propose to tackle coreference resolution through a clustering approach with approximate discriminant.

6 Experimental Results

We have already developed two modeling approaches based on structured perceptron, namely sequence labeling and fixed nodes tree. We have also applied the sequence labeling approach to three tasks: English part-of-speech tagging (POS), English text chunking, and Portuguese text chunking. Furthermore, we have developed a system for Portuguese dependency parsing based on the fixed nodes tree approach. In this section, we summarize the obtained results and compare them to other systems.

We split our report into two subsections. In the first subsection, we compare the proposed entropy-guided feature generation to two other feature generation approaches – namely, manual templates and polynomial kernels. For this comparison, we set equivalent conditions, i.e., same datasets and basic features. In the second subsection, we compare the best performances obtained by our systems to the best performing available systems. In this case, we eventually compare systems that use different basic features, however they are always evaluated on the same dataset.

6.1 Feature Generation Approaches

We argue that entropy-guided feature generation solves a structured ML modeling bottleneck, that is the need to manually create feature templates. However, many authors have proposed successful systems based on manually generated templates. We use the CoNLL'2006 corpus [32] for Portuguese dependency parsing to compare the proposed entropy-guided feature generation method and a state-of-the-art, manually created template set. [22] proposed MSTParser and currently this is the best performing system for DP on the Portuguese corpus from CoNLL'2006. MSTParser is based on 21 manually created feature templates that conjoin basic features in order to form more discriminative, compound features. We use these templates to train a structured perceptron model and evaluate it on the CoNLL'2006 Portuguese corpus. The performance measure for this task is the unlabeled attachment score (UAS) that expresses the percentage of tokens attached to their correct heads. The performances of this system and the system with entropy-guided templates can be found in Table 1 (first row). We can observe that EFG reduces

		Other System			EFG
Task	**Language**	**Algorithm**	**Features**	**F_1**	**F_1**
DP	Portuguese	S-Perceptron	Manual Templates	90.06	90.28
Chunking	English	SVM	Kernel	93.48	94.12
Chunking	Portuguese	S-Perceptron	Kernel	86.70	87.50

Table 1: Feature generation methods comparison

by 2.2% the error of the manual templates system. That is not a great improvement, but it is more than enough to show the value of EFG. Manual templates require the expensive work of a domain expert. On the other hand, EFG is totally automatic.

Another alternative approach to feature templates are kernel functions. More specifically, polynomial kernels can be used to efficiently generate all possible feature combinations. [33] reported that quadratic kernels produce the best performance for English text chunking. Thus, we compare EFG to 2nd-degree polynomial kernels on two text chunking corpora: the CoNLL'2000 English corpus [34] and the Portuguese corpus from [35]. We can see from Table 1 that EFG outperforms the kernel-based approaches, reducing the prediction error by 9.8% for English (second row) and 6% for Portuguese (third row).

These are substantial improvements.

Moreover, the use of kernel functions implies a great increase on training and test time. [36] reported that training time using a quadratic kernel was 34 times greater than using no kernel function for the same English corpus used here. NLP data is usually very sparse, thus such problems imply a very large number of support vectors. Since kernel methods training time is proportional to the number of support vectors, these methods are specially problematic on NLP data.

6.2 State-of-the-art Systems

We have additionally developed another system based on EFG for part-of-speech tagging for the English language. We report on Table 2 the results for English POS tagging on the

		State-of-the-art		EFG	Error
Task	**Language**	**System**	**F_1**	**F_1**	**Reduction**
POS	English	ETL	96.83	96.72	−3.5%
Chunking	English	SVM	94.17	94.12	−0.9%
Chunking	Portuguese	ETL	86.22	87.50	7.3%
DP	Portuguese	MSTParser	91.36	92.66	15.0%

Table 2: State-of-the-art performances comparison

Brown corpus [37], and also for other three widely used corpora. For each corpus, we also report the best known performing system. The last column of the table reports the error reduction achieved by our EFG based systems. Negative values indicate a greater error of our system. We can see that we achieve good results for English POS tagging and state-of-the-art comparable results for English text chunking. Moreover, we substantially reduce the current state-of-the-art error on Portuguese text chunking and dependency parsing. Our best result for DP is achieved by *directly* including two new basic features in the CoNLL'2006 corpus: text chunking and clause information. This result illustrates the *easiness of use* of EFG, when adding new basic features to our modeling. When using manually generated templates, the domain expert has to design new feature templates considering the available features.

7 Concluding Remarks

We propose an automatic feature generation approach for structured learning that is based on the conditional entropy of local decision variables given input basic features. We evaluate this technique on four NLP datasets and experimentally show its value by comparing its performances with two important alternatives to feature generation, namely manual template generation and polynomial kernel functions. Our experimental results show that the proposed entropy-guided feature generation approach outperforms both alternative methods and, furthermore, presents additional advantages. EFG is faster than kernel methods and avoid the overtraining issue, very common in this alternative method. Compared to manually created feature templates, the fact that EFG bypasses domain experts is highly valuable. Furthermore, the developed EFG-based systems present state-of-the-art comparable performances on the evaluated datasets. Moreover, on Portuguese dependency parsing, our system remarkably reduces the previous smallest error by more than 15%.

As future work, we also propose to model two complex NLP tasks that, as far as we know, have never been approached by structured learning methods before. Namely, quotation extraction and coreference resolution. We plan to model different discriminant problems for these two problems and also for more basic problems. For instance, text chunking and named entity recognition are two tasks that have been recurrently recast as sequence tagging problems, even not being so. The aim, on both tasks, is to identify chunks of words that have a specific meaning in the sentence. Another proposed contribution of this work consists in modeling these two tasks through the weighted interval scheduling problem. In this way, we expect to use more powerful features.

References

[1] NIVRE, J.; HALL, J.; NILSSON, J.; ERYIĞIT, G. ; MARINOV, S.. **Labeled pseudo-projective dependency parsing with support vector machines**. In: PROCEEDINGS OF THE TENTH CONFERENCE ON COMPUTATIONAL NATURAL LANGUAGE LEARNING, CoNLL-X '06, p. 221–225, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[2] WESTON, J.; WATKINS, C.. **Multi–class support vector machines**. Technical Report CSD-TR-98-04, Department of Computer Sciences, Royal Holloway, University of London, 1998.

[3] COLLINS, M.. **Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms**. In: PROCEEDINGS OF THE ACL-02 CONFERENCE ON EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, p. 1–8, 2002.

[4] ALTUN, Y.; TSOCHANTARIDIS, I. ; HOFMANN, T.. **Hidden Markov support vector machines**. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2003.

[5] JOACHIMS, T.. **Learning to align sequences: A maximum-margin approach**. Technical report, Cornell University, 2003.

[6] MCDONALD, R.; CRAMMER, K. ; PEREIRA, F.. **Online large-margin training of dependency parsers**. In: PROCEEDINGS OF THE 43RD ANNUAL MEETING ON ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, ACL'05, p. 91–98, 2005.

[7] TARJAN, R. E.. **Finding optimum branchings**. Networks, 7:25–25, 1977.

[8] CHU, Y. J.; LIU, T. H.. **On the shortest arborescence of a directed graph**. Science Sinica, 14:1396–1400, 1965.

[9] EDMONDS, J.. **Optimum branchings**. Journal of Research of the National Bureau of Standards, 71B:233–240, 1967.

[10] CRAMMER, K.; SINGER, Y.. **Ultraconservative online algorithms for multiclass problems**. Journal of Machine Learning Research, 3:951–991, 2003.

[11] TSOCHANTARIDIS, I.; HOFMANN, T.; JOACHIMS, T. ; ALTUN, Y.. **Support vector machine learning for interdependent and structured output spaces**. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2004.

[12] NOVIKOFF, A. B.. **On convergence proofs on perceptrons**. In: PROCEEDINGS OF THE SYMPOSIUM ON THE MATHEMATICAL THEORY OF AUTOMATA, 1962.

[13] COLLINS, M.. **Ranking algorithms for named-entity extraction: Boosting and the voted perceptron**. In: PROCEEDINGS OF THE ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 2002.

[14] TASKAR, B.; GUESTRIN, C. ; KOLLER, D.. **Max–margin Markov networks**. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 2004.

[15] VAPNIK, V.. **Statistical Learning Theory**. Wiley, 1998.

[16] BRILL, E.. **Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging**. Comput. Linguist., 21:543–565, Dec. 1995.

[17] RAMSHAW, L.; MARCUS, M.. **Text chunking using transformation-based learning**. In: PROCEEDINGS OF THE THIRD WORKSHOP ON VERY LARGE CORPORA, p. 82–94, 1995.

[18] MILIDIÚ, R. L.; DUARTE, J. C. ; CAVALCANTE, R.. **Machine learning algorithms for portuguese named entity recognition**. In: FOURTH WORKSHOP IN INFORMATION AND HUMAN LANGUAGE TECHNOLOGY, 2006.

[19] DOS SANTOS, C. N.; MILIDIÚ, R. L.. **Entropy guided transformation learning**. In: FOUNDATIONS OF COMPUTATIONAL INTELLIGENCE (1), p. 159–184. Springer, 2009.

[20] KUDO, T.; MATSUMOTO, Y.. **Chunking with support vector machines**. In: PROCEEDINGS OF THE SECOND MEETING OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS ON LANGUAGE TECHNOLOGIES, NAACL'01, p. 1–8, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.

[21] CARRERAS, A. X.; MÀRQUEZ, B. L. ; CASTRO, C. J.. **Filtering-ranking perceptron learning for partial parsing**. Machine Learning, 60:41–71, 2005. 10.1007/s10994-005-0917-x.

[22] MCDONALD, R.; LERMAN, K. ; PEREIRA, F.. **Multilingual dependency analysis with a two-stage discriminative parser**. In: IN PROCEEDINGS OF THE CONFERENCE ON COMPUTATIONAL NATURAL LANGUAGE LEARNING (CONLL, p. 216–220, 2006.

[23] ISOZAKI, H.; KAZAWA, H.. **Efficient support vector classifiers for named entity recognition**. In: PROCEEDINGS OF THE 19TH INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS - VOLUME 1, COLING'02, p. 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[24] QUINLAN, J. R.. **C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)**. Morgan Kaufmann, 1 edition, 1992.

[25] SU, J.; ZHANG, H.. **A fast decision tree learning algorithm**. In: PROCEEDINGS OF THE 21ST NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, p. 500–505, 2006.

[26] DIETTERICH, T.. **Machine learning for sequential data: A review**. In: PROCEEDINGS OF THE JOINT IAPR INTERNATIONAL WORKSHOP ON STRUCTURAL, SYNTACTIC, AND STATISTICAL PATTERN RECOGNITION, 2002.

[27] PRADHAN, S.; RAMSHAW, L.; MARCUS, M.; PALMER, M.; WEISCHEDEL, R. ; XUE, N.. **Conll-2011 shared task: Modeling unrestricted coreference in ontonotes**. In: PROCEEDINGS OF THE FIFTEENTH CONFERENCE ON COMPUTATIONAL NATURAL LANGUAGE LEARNING SHARED TASK, p. 1–27, Portland, USA, 2011. ACL.

[28] MCDONALD, R.; PEREIRA, F.. **Online learning of approximate dependency parsing algorithms**. In: IN PROC. OF EACL, p. 81–88, 2006.

[29] COLLINS, M.; ROARK, B.. **Incremental parsing with the perceptron algorithm**. In: PROCEEDINGS OF THE 42ND ANNUAL MEETING ON ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, ACL'04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[30] MOORE, R. C.. **A discriminative framework for bilingual word alignment**. In: PROCEEDINGS OF THE CONFERENCE ON HUMAN LANGUAGE TECHNOLOGY AND EMPIRICAL METHODS IN NATURAL LANGUAGE PROCESSING, HLT'05, p. 81–88, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[31] DAUMÉ, III, H.; MARCU, D.. **Learning as search optimization: approximate large margin methods for structured prediction**. In: PROCEEDINGS OF THE 22ND INTERNATIONAL CONFERENCE ON MACHINE LEARNING, ICML'05, p. 169–176, New York, NY, USA, 2005. ACM.

[32] BUCHHOLZ, S.; MARSI, E.. **CoNLL-x shared task on multilingual dependency parsing**. In: PROCEEDINGS OF THE TENTH CONFERENCE ON NATURAL LANGUAGE LEARNING, p. 149–164, 2006.

[33] KUDO, T.; MATSUMOTO, Y.. **Fast methods for kernel-based text analysis**. In: PROCEEDINGS OF THE 41ST ANNUAL MEETING ON ASSOCIATION FOR COMPUTATIONAL LINGUISTICS - VOLUME 1, ACL'03, p. 24–31, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[34] TJONG KIM SANG, E. F.; BUCHHOLZ, S.. **Introduction to the conll-2000 shared task: chunking**. In: PROCEEDINGS OF THE 2ND WORKSHOP ON LEARNING LANGUAGE IN LOGIC AND THE 4TH CONFERENCE ON COMPUTATIONAL NATURAL LANGUAGE LEARNING - VOLUME 7, ConLL '00, p. 127–132, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[35] FERNANDES, E. R.; DOS SANTOS, C. N. ; MILIDIÚ, R. L.. **A machine learning approach to portuguese clause identification**. In: Pardo, T. A. S.; Branco, A.; Klautau, A.; Vieira, R. ; de Lima, V. L. S., editors, PROCEEDINGS OF THE 9TH INTERNATIONAL CONFERENCE ON COMPUTATIONAL PROCESSING OF THE PORTUGUESE LANGUAGE, volumen 6001 de **Lecture Notes in Computer Science**, p. 55–64. Springer, 2010.

[36] WU, Y.-C.; CHANG, C.-H. ; LEE, Y.-S.. **A general and multi-lingual phrase chunking model based on masking method**. In: Gelbukh, A., editor, COMPUTATIONAL LINGUISTICS AND INTELLIGENT TEXT PROCESSING, volumen 3878 de **Lecture Notes in Computer Science**, p. 144–155. Springer Berlin / Heidelberg, 2006.

[37] FRANCIS, W. N.; KUCERA, H.. **Frequency analysis of english usage: Lexicon and grammar**. Houghton Mifflin, Boston, 1982.