3
Entropy-Guided Feature Generation

A task dataset includes features that usually are either (i) naturally included in the very phenomenon of interest, like words in NLP tasks; (ii) simply derived from other basic features, like word capitalization patterns; or (iii) automatically generated by external systems, like part-of-speech tags. We denote this dataset-provided information as *basic features*. At the same time, most structure learning algorithms are based on linear models, since such algorithms have strong theoretical guarantees regarding their prediction performance and, moreover, are computationally efficient. However, linear models on basic features alone do not capture all the relevant relationships among these variables.

For instance, dependency parsing is highly non-linear on the basic features. One important basic feature for DP is the POS of words. By observing the example in Figure 1.1, one can notice that both nouns system and result are modifiers of the verb achieves. This is a very strong pattern in DP and, thus, the binary feature `modifier is a noun AND head is a verb` is very important. This feature is a conjunction of two basic features, namely the modifier POS tag and the head POS tag. On the other hand, taking each of these basic features independently is not informative, since most POS tags can be head or modifier of many dependencies, depending on the context. For instance, the nouns system and results, that modify the main verb, are also heads of the pronoun Our and the adjective best, respectively.

Conjoining basic features to derive new features is a common way to introduce nonlinear contextual patterns into linear models. Frequently, a domain expert manually generates feature templates by conjoining the given basic features in order to capture discriminative contextual patterns. MSTParser, for instance, uses 21 feature templates that were manually created from basic features. These templates include from one to six features, indicating that the trained model is highly non-linear on the basic input features.

In this chapter, we describe the proposed entropy-guided feature generation method for structure learning. EFG automatically derives a set of basic feature conjunctions, which we denote *feature templates*. These templates

are later used to generate the *derived features*, which comprise the joint feature vectors $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})$ used in the structured modeling described earlier. EFG conjoins basic features that are useful to predict *local* variables.

3.1 Basic Dataset

As seen in the previous chapters, in dependency parsing, features are functions of dependency tree edges. Given an edge $e = (i, j)$ linking token x_i to token x_j, let us examine only its *categorical* basic features. Assume there are K basic features given by the vector $\boldsymbol{\Psi}(e) = (\psi_1(e), \ldots, \psi_K(e))$. For $k = 1, \ldots, K$, we have that $\psi_k(e) \in X_k$, where X_k is the finite set of possible values for the basic feature ψ_k. Additionally, we associate each edge $e = (i, j)$ with a binary label $y(e)$, such that $y(e) = 1$ if x_i is the head of x_j and $y(e) = 0$ otherwise.

Using all edges in the training set $\mathcal{D}$, we obtain the *basic dataset* $D = \{(\boldsymbol{\Psi}(e), y(e))\}$ comprising the basic feature vectors of edges along with their binary labels. In Table 3.1, we depict an example of such a dataset for the sentence in Figure 2.1. This example includes the following basic features: head-word is the word of the head token x_i; mod-word is the word of the modifier token x_j; head-pos is the POS tag of x_i; mod-pos is the POS tag of x_j; dist is the distance between x_i and x_j in tokens; and side is the side of x_j in relation to x_i.

e		$\boldsymbol{\Psi}(e)$						$y(e)$
i	j	**head-word**	**mod-word**	**head-pos**	**mod-pos**	**dist**	**side**	
0	1	*root*	John	*root*	noun	1	right	0
0	2	*root*	saw	*root*	verb	2	right	1
0	3	*root*	Mary	*root*	noun	3	right	0
1	2	John	saw	noun	verb	1	right	0
1	3	John	Mary	noun	noun	2	right	0
2	1	saw	John	verb	noun	1	left	1
2	3	saw	Mary	verb	noun	1	right	1
3	2	Mary	saw	noun	verb	1	left	0

Table 3.1: Basic dataset for the sentence in Figure 2.1.

The entropy guided feature generation method automatically generates feature templates for a structure learning problem by conjoining basic features that are highly discriminative together. EFG is based on the conditional entropy of the local decision variables $y(e)$ given the basic features $\boldsymbol{\Psi}(e)$.

3.2 Conditional Entropy and Information Gain

Entropy is a measure of the uncertainty in a random variable outcome. Given a binary random variable y and the probability $Pr[y = 1] = p$, the

entropy of y is given by

$$H(y) = -p \log_2 p - (1-p) \log_2 (1-p),$$

where $0 \log_2(0)$ is defined to be equal to 0. In Figure 3.1, we plot the entropy $H(y)$ versus p. One can see that the maximum entropy is achieved when $p = 0.5$, that is, when the uncertainty on the outcome of y is maximum.

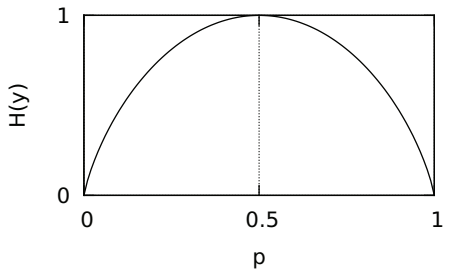

Figure 3.1: Entropy $H(y)$ of a random binary variable y versus $Pr[y = 1]$ that is denoted by p.

Given the basic dataset D, one can easily estimate p as the fraction of edges in which $y(e)$ is equal to 1. Then, the *empirical entropy* of y on D is denoted by $H_D(y)$ and can be calculated using the estimated probability of $y = 1$ on the given dataset. In the following, we simply use the term entropy to denote empirical entropy.

Considering the basic dataset D, we define the *conditional entropy* of y given the basic feature ψ_k as

$$H_D(y|\psi_k) = \sum_{\sigma \in X_k} \frac{|D_{k,\sigma}|}{|D|} \cdot H_{D_{k,\sigma}}(y),$$

where $D_{k,\sigma}$ is the subset of edges in D whose feature ψ_k is equal to σ, that is, $D_{k,\sigma} = \{(\mathbf{\Psi}(e), y(e)) \in D \mid \psi_k(e) = \sigma\}$. The conditional entropy $H_D(y|\psi_k)$ is the entropy of y on D when an additional information (feature ψ_k) is given. From Gibb's inequality, it follows that $H_D(y|\psi_k) \leq H_D(y)$, that is, the knowledge of any additional information can only reduce uncertainty.

Additionally, the *information gain* (IG) of ψ_k on D is given by

$$IG_D(\psi_k) = H_D(y) - H_D(y|\psi_k),$$

which corresponds to the reduction on the entropy of y if feature ψ_k is known. Hence, IG helps to select high discriminative features with respect to a target variable. It is straightforward to generalize information gain to

a subset of features. Thus, we have a valuable metric to measure nonlinear feature conjunctions and select the most informative ones.

Unfortunatelly, to analyse all possible feature conjunctions is practically infeasible and, moreover, to find the best conjunctions is an NP-complete problem (Hyafil and Rivest, 1976). On the other hand, decision tree (DT) learning provides a simple yet effective algorithm that generates different subsets of informative features, greedily guided by some informativeness metric. The most popular DT algorithms (Quinlan, 1992; Su and Zhang, 2006) use information gain as that metric. Therefore, we use decision tree induction to generate feature combinations that are highly discriminative together.

3.3 Decision Tree Learning

Decision tree 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 given by the prediction variable values. Usually, after the tree is grown, a pruning step is carried out in order to avoid overfitting. In Figure 3.2, we present a decision tree learned from a basic dataset. Each *internal* node in the DT corresponds to a feature, each *leaf* node has a label value (0 or 1, in the binary case), and each edge is labeled with a value of the source node feature.

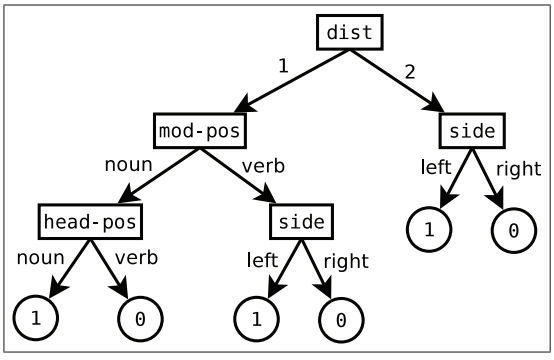

Figure 3.2: A decision tree.

The most popular decision tree learning algorithms (Quinlan, 1992; Su and Zhang, 2006) use information gain to select the most informative feature. Hence, they provide a quick way to obtain entropy guided feature selection.

We propose a new automatic feature generation method for structure learning algorithms. The key idea is to use decision tree induction to conjoin the basic features. One of the most used algorithms for DT induction is C4.5 (Quinlan, 1992). We use Quinlan's C4.5 system to obtain the required entropy guided selected features.

3.4 Feature Templates

The first step of the proposed method is to train a decision tree on the basic dataset. For dependency parsing, the decision variable indicates whether an edge links a token to its corresponding head token. We use the edges of all training examples, that is, for each training sentence, we generate an example for each candidate edge. Thus, the learned DT predicts whether an edge corresponds to a correct dependency or not.

Our method uses a very simple decomposition scheme to extract feature templates. This decomposition is based on a depth-first traversal of the learned DT and is recursively defined as follows. For each *internal* node that is visited, a new template is created by conjoining the node feature with its parent template. Since we aim to generate feature *templates* – conjunctions of basic features not including feature values – we disconsider the feature values and the decision variable values in the DT. Thus, we do not make use of edge labels nor leaf nodes. Figure 3.3 illustrates our method. The tree in the left side of this figure is the skeleton obtained from the decision tree in Figure 3.2 by discarding the aforementioned pieces of information. Then, it remains a tree whose nodes are basic features with high discriminative power. The generated templates are listed in the right side of the figure. In other words, we create a template with the features in each path from the root node to every other internal node in the given decision tree. Additionally, we eliminate template duplicates.

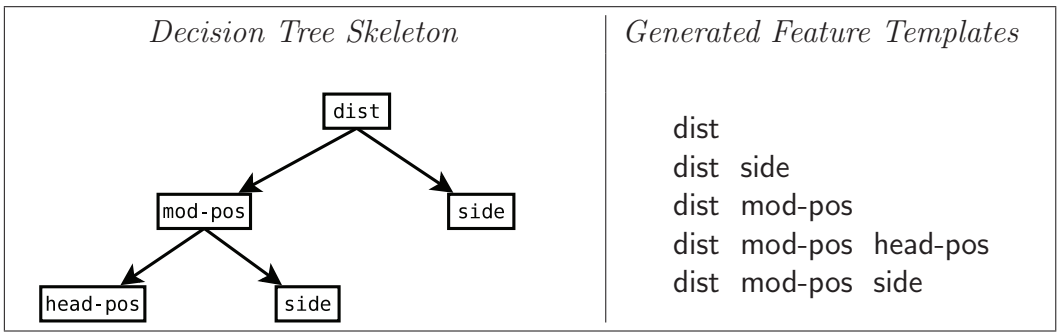

Figure 3.3: Feature template induction from a decision tree.

Since the DT learning algorithm greedily chooses the feature with the

highest information gain at each step, our method generates feature templates with high discriminative power based on entropy. This method is able to provide a very large number of templates. Hence, to limit the maximum template length, we use C4.5 pruned trees and additionally limit the maximum template length when traversing the DT. This parameter is clearly task dependent and must be calibrated by cross-validation or by means of a development set.

3.5 Generated Features

Finally, we employ the generated templates to induce all binary contextual features that occur in the structured dataset $\mathcal{D}$. For each template, we generate several binary features, each one corresponding to the assignment of valid values to the template features. These derived features comprise the structured model feature vectors $\mathbf{\Phi}(\boldsymbol{x}, i, j) = (\phi_1(\boldsymbol{x}, i, j), \ldots, \phi_M(\boldsymbol{x}, i, j))$. For instance, one of the derived features for the dist mod-pos side template in Figure 3.3 is given by

$$\phi_m(\boldsymbol{x}, i, j) = \begin{cases} 1 & \text{if dist=2 and mod-pos=noun and side=left,} \\ 0 & \text{otherwise.} \end{cases}$$

Observe that this feature captures a context that is not used by the DT in Figure 3.2. Indeed, we drop the DT feature values when generating the templates and then instantiate these templates based on every context that occurs in the dataset.

3.6 Empirical Results

In this section, we compare the performances of systems based on the proposed EFG method to a system based on the best available set of manual templates for dependency parsing. For this task, our systems do not make use of second- or third-order features. Thus, we use only first-order features to perform this comparison. In Table 3.2, we show the performance of an SPerc system using the best available manual templates along with the performances of two systems based on EFG. First, let us focus on a comparison of EFG to manual templates under same conditions, i.e., same basic features, same learning algorithm, and same datasets. We use the SPerc learning algorithm and the first-order features provided in the templates from McDonald et al. (2006). The first row of Table 3.2 shows the performance when using the

Basic Features	**Feature Generation**	**UAS**
1st order	Manual	90.06
1st order	EFG	90.28
1st+ck+clause	EFG	92.66

Table 3.2: Performances of EFG and manual templates on the Portuguese CoNLL-2006 dependency parsing dataset.

manual templates; and, the second row presents the performance when using EFG to automatically generate non-linear features. EFG outperforms manual templates by 0.22 on UAS. This is not a very big improvement, but shows that EFG is able to automatically generate complex feature templates that are competitive, and even better, than state-of-the-art templates that require substantial human effort.

Fernandes et al. (2010b) present text chunking and clause identification for the Bosque corpus, which comprises the Portuguese CoNLL-2006 dataset. We provide these two basic pieces of information as basic features to the EFG method, as well as the basic features used earlier, and train an SPerc model using the provided templates. The performance obtained by this system is shown in the last row of Table 3.2. We can see that we achieve an impressive improvement around 2.4% on UAS. Moreover, this improvement is achieved by simply including two basic features, without any human effort, as would be required if one used manual templates.