3
Machine Learning Algorithms

In this chapter, we present the ML algorithms used to tackle the Quotation Extraction task. We use *Entropy Guided Transformation Learning* (ETL) (9) and *Structured Perceptron* (6). ETL is used to predict a label sequence and uses information about tokens in a token neighbourhood. As we can model our task as a label sequence problem and ETL has been successfully applied in several NLP tasks (10, 14, 25, 16, 11, 26, 29), we think it is a good choice. The Structured Perceptron, since it is a structured learning method, is used to predict complex and interdependent outputs like sequences, trees, and even more general graphs. As we can model our task as a prediction of quotations and their authors given a piece of news and structured learning methods have been successfully applied in several NLP tasks (1, 6, 4, 40, 22), we think it is a good choice.

PUC-Rio - Certificação Digital Nº 1012671/CA

3.1
Entropy Guided Transformation Learning

ETL is an evolution of *Transformation Based Learning* (TBL) (3) that solves the main bottleneck of TBL, which is the automatic generation of template sets. In this section, we present the TBL and ETL algorithms.

3.1.1
Transformation Based Learning

TBL is a ML algorithm for classification tasks. It has been used for several NLP tasks such as part-of-speech tagging (3, 17, 11), phrase chunking (31, 24, 12, 27), spelling correction (23), appositive extraction (19), named entity recognition (18, 28) and semantic role labelling (20). TBL learns transformation rules which correct mistakes in the corpus. The following rules illustrates three transformation rules applied in the POS tagging for the Portuguese task.

```
pos[0]=ART pos[1]=ART -> pos[0]=PREP
pos[0]=ART pos[1]=V word[0]=a -> pos[0]=PREP
pos[0]=N pos[-1]=N pos[-2]=ART -> pos[0]=ADJ
```

The rules above check the features pos[0], the POS of the current word; pos[1], the POS of the next word; pos[-1], the POS of the previous word; pos[-2], the POS of the word two before; word[0], the current word. We should read the first rule as

"**IF** the POS of the current word is an *article*
AND the POS of the next word is an *article*
THEN change the POS of the current word to *preposition*."

TBL rules are composed by the left hand side and the right hand side. The former is a conjunction of *feature=value* tests. The latter is a value assignment for the target feature. Those rules follow patterns which specify possible feature combinations in the left hand side, what we call *rule templates*. A template set defines the candidate rule space to be searched.

TBL receives the correctly labelled training set as an input; which is a baseline system (BLS) that provides a initial labelling for training examples, which is usually based on simple statistics of the training corpus; a set of rule templates.

PUC-Rio - Certificação Digital Nº 1012671/CA

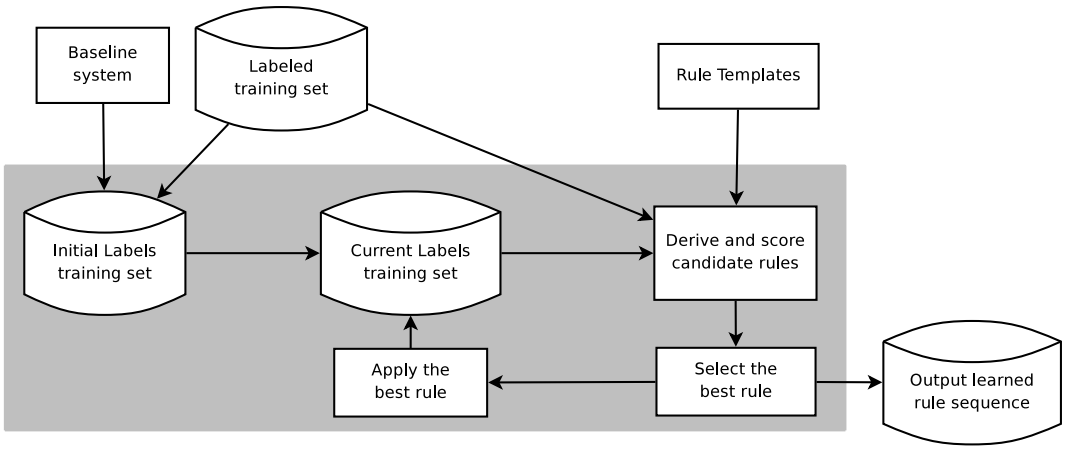

Figure 3.1: TBL algorithm

We show TBL algorithm in Figure 3.1. It greedily learns rules that reduce the BLS classification errors. At each interaction, it learns the highest score rule. The score of a rule r is the difference between the repaired errors by r and the created errors by r. The learning process stops when the score is smaller than a threshold, which is a TBL parameter.

TBL templates capture relevant feature combinations. They are created by domain experts. Therefore, the model quality strongly depends on the expert's skills. For instance, Ramshaw and Marcus (31) propose a set of 100 templates for the phrase chunking task. Florian (18) proposes a set of 133

templates for the named entity recognition task. Higgins (20) proposes a set of 130 templates for the semantic role labeling task. Thus human derived templates are the bottleneck for the effective use of TBL.

3.1.2 Entropy Guided Transformation Learning

ETL is a ML algorithm used for classification tasks. ETL solves the main bottleneck of TBL, which is the automatic generation of template sets. It uses Information Gain (IG) in order to select feature combinations that provide good template sets. We illustrate the ETL algorithm in Figure 3.2.

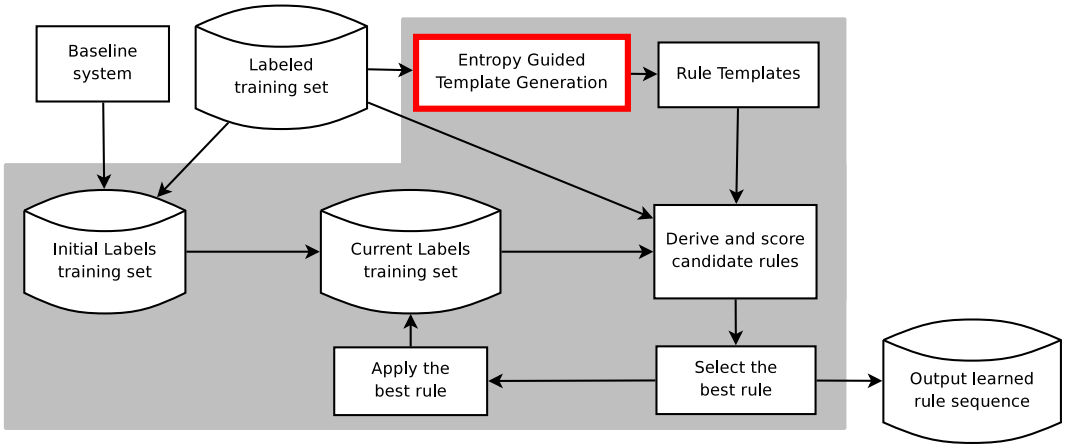

Figure 3.2: ETL algorithm

The IG is based on data entropy, which is a key measure for many feature selection strategies. It is used by the most popular Decision Tree (DT) induction algorithms.

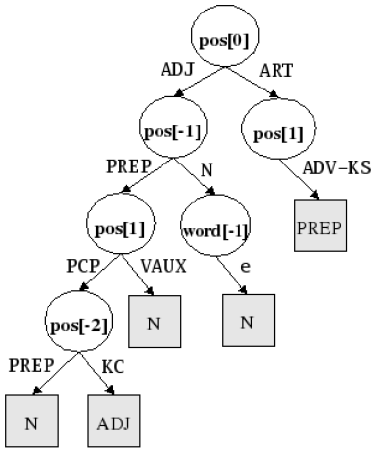

Figure 3.3: Decision tree

PUC-Rio - Certificação Digital Nº 1012671/CA

ETL uses DT induction algorithms to automatically generate template sets. The DT induction algorithm recursively partitions the training set using the feature which provides the largest IG. The result is a DT, where the nodes are the selected features and the arcs are the selected feature values. In Figure 3.3, we show an example of a DT generated by a DT induction algorithm.

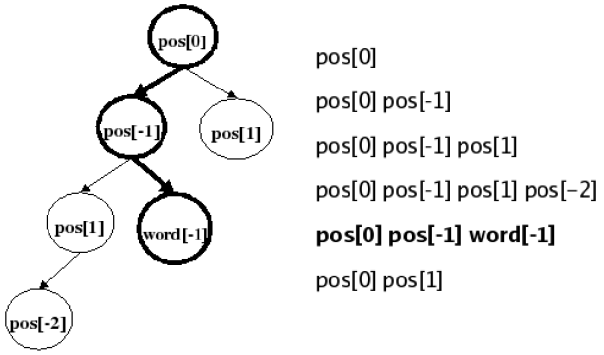

Figure 3.4: Decision tree after the elimination process and the extracted templates.

ETL takes the generated DT and eliminates leaves and arc labels. Then ETL visits all tree nodes and, for each node, it creates a template that combines features in the path from root to node. In Figure 3.4, we present the derived decision tree and the generated templates. These templates are passed to TBL. Thus, ETL substitutes the need of human experts when generating the template set.

3.2 Structured Perceptron

The Structured Perceptron algorithm is analogous to the well-known Perceptron algorithm (32). Given a training set $\mathcal{D}$, composed of pairs $(\boldsymbol{x}, \boldsymbol{y})$ of correct inputs-outputs, both with complex structures, the algorithm updates model $\boldsymbol{w}$ in order to improve its quality. At each iteration, a training instance $(\boldsymbol{x}, \boldsymbol{y})$ is drawn from $\mathcal{D}$ and two major steps are performed: prediction using the current model and model update based on the difference between the correct and the predicted outputs. In Figure 3.5, we present the Structured Perceptron algorithm.

We need to define a predictor $F(\boldsymbol{x})$, whose objective is to find a $\hat{\boldsymbol{y}}$ which maximizes the dot product between model $\boldsymbol{w}$ and feature vector $\boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}})$.

PUC-Rio - Certificação Digital Nº 1012671/CA

1. $\boldsymbol{w} \leftarrow \mathbf{0}$
2. while no convergence
3. for each $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$
4. $\hat{\boldsymbol{y}} \leftarrow \arg\max_{\hat{\boldsymbol{y}} \in \mathcal{Y}(\boldsymbol{x})} \langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}}) \rangle$
5. $\boldsymbol{w} \leftarrow \boldsymbol{w} + \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) - \boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}})$
6. endfor
7. endwhile
8. return $\boldsymbol{w}$

Figure 3.5: Structured Perceptron algorithm

Predictor $F(\boldsymbol{x})$ is an optimization problem whose objective function is linear in the feature vector $\boldsymbol{\Phi}(\boldsymbol{x}, \hat{\boldsymbol{y}})$.

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

Thus, the learning problem consists of determining $\boldsymbol{w}$, such that the resulting predictor $F(\boldsymbol{x})$ is accurate on the training data and, moreover, shows good generalization performance on unseen data.

3.2.1 Prediction Problem

As we have seen in the previous section, we need to define a predictor $F(\boldsymbol{x})$, which is an optimization problem. The decomposition of $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y})$ in $\mathcal{Y}(\boldsymbol{x})$ defines the prediction problem. The prediction problem is task-dependant and model-dependant. For instance, for multiclass tasks or tasks which have a reduced search space, the prediction problem is the enumeration of all possible solutions and, subsequently, the choice of the best one. For sequence labeling tasks, the prediction problem can be solved using the Shortest Path algorithm. For branching tasks, i.e. dependency parsing, the prediction problem is casted in a maximum branching problem, which is solved using a maximum branching algorithm (38).

The task of Quotation Extraction is to find all quotations and their authors, given a piece of news. We cast our task in a well-known optimization problem. We present that problem and its respective algorithm in Chapter 4.

PUC-Rio - Certificação Digital Nº 1012671/CA

3.2.2 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 for unseen data. The MIRA (7) is a generalization of the Structured Perceptron algorithm which 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 (39, 13). For a training instance $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$, instead of the ordinary discriminant problem $F(\boldsymbol{x})$, we use a *loss-augmented* discriminant problem in step (4) of the Structured Perceptron algorithm that measures the difference between the predicted output $\hat{\boldsymbol{y}}$ and the correct output $\boldsymbol{y}$.

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

where $\ell(\cdot, \cdot) \geq 0$ is a given loss function that measures the difference between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$.

3.2.3 Entropy Guided Feature Generation

Feature generation is an important subtask for structured learning modeling. Usually, a task dataset includes some basic features that are either naturally included in the very phenomenon of interest, such as words for NLP tasks; simply derived from other basic features, such as capitalization information; or automatically generated by external systems, such as 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. However, the quality of the model depends on the expert's skills as we see in Section 3.1.

Therefore, we use a method to automatically generate the feature template set (15). This method is analogous to the ETL algorithm. The only difference is how a DT example is built. As we classify a token sequence in ETL, the DT example is composed of features relative to a specific token, including tokens in its neighbourhood. In Structured Perceptron, the DT example is composed of features related to a tuple candidate.

PUC-Rio - Certificação Digital Nº 1012671/CA