5
Prediction Problems

In this chapter, we discuss prediction problems, a component that plays a central role in the SL framework. The prediction problem for an input $\boldsymbol{x}$ in the SL formulation has the form

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

This is a very general form for an optimization problem. In fact, the only requirement is that the objective function be linear on some feature representation. On the other hand, the joint feature representation has no explicit restriction and the output space is also arbitrary. Thus, ESL is just a framework for learning parameters for general linear predictors. And, there are several learning algorithms for this framework with strong guarantees regarding both prediction performance and learning time. In Table 5.1, we present a list of SL problems along with the corresponding output structures and prediction problems.

Task	Output Structure	Prediction Problem
Dependency parsing	Rooted tree	Maximum branching
Part-of-speech tagging	Sequence	Longest path on DAG
Text chunking	Sequence	Longest path on DAG
Quotation extraction	Segmentation	Weighted interval scheduling
Coreference resolution	Clustering	*Latent* maximum branching

Table 5.1: List of tasks and the corresponding output structures and prediction problems.

The output space $\mathcal{Y}(\boldsymbol{x})$ is represented by task-specific hard constraints that are embedded in the prediction algorithm. Hence, $\mathcal{Y}(\boldsymbol{x})$ can comprise any constraint that is efficiently handled by the optimization algorithm. For most SL problems, these constraints are difficult to be learned from data; or, at least, it is unnecessary to do so, since they are never violated in any example. For instance, in dependency parsing, the learning algorithm expends no effort on estimating parameters to avoid cycles in the output structure. The prediction algorithm never extracts cycles because it is constrained to extract only trees. This is a very flexible way to represent the valid prediction outputs.

Additionally, it allows the modeler to make use of countless theoretical and practical results from combinatorial optimization.

The second arbitrary component in the prediction problem is the joint feature vector representation $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) = (\phi_1(\boldsymbol{x}, \boldsymbol{y}), \ldots, \phi_M(\boldsymbol{x}, \boldsymbol{y}))$. Each feature function $\phi_m(\boldsymbol{x}, \boldsymbol{y})$ is called global feature because it gives a value regarding the whole output structure $\boldsymbol{y}$. However, global features are usually factored along the output structure and, consequently, the scoring function $\langle \boldsymbol{w}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle$ is factored in the same way. Otherwise, the prediction algorithm would need to enumerate all possible outputs to determine the best scoring one. In multiclass classification, for instance, there is no feature factorization and the prediction algorithm just enumerates all classes, computes their scores and picks the highest scoring one. That is feasible when the number of classes is limited, which is the case for multiclass classification. The feature factorization defines the dependencies among the output variables. For dependency parsing, features are factored on dependency edges (i, j) and a tree score is given just by independently summing the scores of its edges. Thus, the prediction problem is equivalent to the maximum branching problem.

In the next sections, we briefly describe some important structures along with the proposed feature factorizations and the resulting prediction algorithms. These aspects are discussed later in more details.

5.1 Rooted Tree

Dependency parsing consists in predicting a rooted tree underlying a given sentence. The nodes of this tree are fixed: the sentence tokens. Let $\boldsymbol{x} = (x_0, x_1, \ldots, x_N)$ be an input sequence, where x_t is the t-th token and x_0 is a special node that is always the root of the tree. The prediction output space $\mathcal{Y}(\boldsymbol{x})$ is the set of all rooted trees whose nodes are tokens in $\boldsymbol{x}$ and the root node is x_0.

In this work, we use the feature factorization proposed by McDonald et al. (2005), which defines features over independent edges. In this case, the prediction problem is efficiently solved by the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967). However, more complex models are possible. McDonald and Pereira (2006) propose features that depend on more than two tokens. More specifically, they use the so called second-order features that depend on two dependency edges (i, j) and (i, k). Koo et al. (2010) further extends this model by introducing third-order features that also depend on two edges, but include dependencies on grandparent tokens. They show significant improvements on parsing performance by including

high-order features. However, the complexity of the prediction algorithm also grows. In fact, the prediction problem becomes NP-hard when these features are considered. Thus, they solve the problem by approximation algorithms.

5.2 Sequence Labeling

Sequence labeling Dietterich (2002) is to find a sequence of labels $\boldsymbol{y} = (y_1, \ldots, y_N)$, where $y_t \in S$, for a given input sequence of tokens $\boldsymbol{x} = (x_1, \ldots, x_N)$. That is, each token x_t is annotated with a label $y_t \in S$, where S is a fixed set of labels. The prediction output space for an input $\boldsymbol{x}$ is simply the set of all possible label sequences with length N, that is $\mathcal{Y}(\boldsymbol{x}) = S^N$.

Collins (2002b) proposes a feature factorization for sequence labeling problems that relies on a Markovian property. The best scoring label for a specific token x_t depends only on the label itself y_t and the previous token label y_{t-1}. In this factorization, there are two types of features: $\boldsymbol{\Phi}^{\text{obs}}(\boldsymbol{x}, y_t)$ are observation features that depend only on the input $\boldsymbol{x}$ and individual token labels; and $\boldsymbol{\Phi}^{\text{trans}}(y_{t-1}, y_t)$ are transition features that depend on two consecutive labels. The resulting prediction problem is reduced to the longest path problem on a directed acyclic graph, which can be efficiently solved by a dynamic programming algorithm.

5.3 Sequence Segmentation

Given a document, quotation extraction is to identify quotes and, additionally, to associate each quote to its author. For a training example $(\boldsymbol{x}, \boldsymbol{y})$, the input $\boldsymbol{x}$ is composed by a set of K candidate authors $\boldsymbol{a} = \{a_1, \ldots, a_K\}$ and a set of N candidate quotes $\boldsymbol{q} = \{q_1, \ldots, q_N\}$, where each quote corresponds to a segment of the input document. The set of candidate quotes can overlap each other, but the correct quotes do not. The output space is thus all subsets of non-overlapping candidate quotes such that each selected quote is associated to exactly one author.

Fernandes (2012) proposes a structure learning modeling for quotation extraction in which features depend on the association of a quote to an author. In that way, the prediction problem is to find non-overlapping segments associated to authors whose weights are maximum. This problem is equivalent to the weighted interval scheduling for which there is an efficient dynamic programming algorithm.

5.4 Clustering

Coreference resolution Pradhan et al. (2011) consists in identifying mentions to real-world entities in a document and clustering mentions that refer to the same entity. This task is usually split into two subtasks: mention detection and mention clustering. Mention detection is easily performed by recovering all noun phrases in the document. The most interesting task is mention clustering. The prediction output space for this task comprises all possible clustering of the given mentions. The number of clusters is unknown.

Usually, coreference systems use features that depend on pairs of mentions. We follow this idea, but we introduce a novel modeling for coreference resolution. We assume that an entity cluster is represented by a rooted tree, denoted *coreference tree*. A directed edge (m_i, m_j) from mention m_i to mention m_j in this tree indicates that m_j is a reference to the more general mention m_i. In that way, we model the prediction problem as a maximum branching problem on the graph whose nodes are the given set of mentions.