

Cícero Nogueira dos Santos

Entropy Guided Transformation Learning

D.Sc. Thesis

Thesis presented to the Graduate Program of the Department of Informatics, PUC-Rio, in partial fulfillment of the requirements for the degree of Doctor of Science in the subject of Informatics.

Advisor: Prof. Ruy Luiz Milidiú

Rio de Janeiro
March 2009



Cícero Nogueira dos Santos

Entropy Guided Transformation Learning

Thesis presented to the Graduate Program of the Department of Informatics, PUC–Rio, in partial fulfillment of the requirements for the degree of Doctor of Science in the subject of Informatics. Approved by the following commission.

Prof. Ruy Luiz Milidiú

Advisor

Departamento de Informática — PUC–Rio

Prof. Daniel Schwabe

Departamento de Informática — PUC–Rio

Prof. Violeta de San Tiago Dantas Barbosa Quental

Departamento de Letras — PUC–Rio

Prof. Fernando Antonio de Carvalho Gomes

Departamento de Computação — UFC

Prof. Bianca Zadrozny

Departamento de Ciência da Computação — UFF

Prof. Raúl Pierre Renteria

Departamento de Informática — PUC–Rio

Prof. José Eugenio Leal

Coordinator of the Centro Técnico Científico — PUC–Rio

Rio de Janeiro — March 20, 2009

All rights reserved. No part of this thesis may be reproduced in any form or by any means without prior written permission of the University, the author and the advisor.

Cícero Nogueira dos Santos

Cícero Nogueira dos Santos graduated in 2003 from the Universidade Regional do Cariri (URCA) in Production Engineering. In 2005, he obtained a Master degree in Computer Science at the Instituto Militar de Engenharia (IME).

Cataloging Data

Santos, Cícero Nogueira dos

Entropy guided transformation learning / Cícero Nogueira dos Santos; advisor: Ruy Luiz Milidiú. – 2009.

85 f: il. ; 30 cm

Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2009.

Inclui bibliografia.

1. Informática – Teses. 2. Aprendizado de transformações guiado por entropia. 3. Aprendizado baseado em transformações. 4. Etiquetagem morfossintática. 5. Identificação de sintagmas. 6. Reconhecimento de entidades nomeadas. 7. Etiquetagem de papéis semânticos. I. Milidiú, Ruy Luiz. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

First of all, I would like to express my deepest sense of gratitude to my advisor Prof. Ruy Luiz Milidiú for his support, friendship, encouragement and excellent guidance throughout my PhD study.

I would like to express my gratitude to CNPq for the financial support, without which this work would not have been realized.

I am thankful to the PUC–Rio’s Postgraduate Program in Informatics for providing an excellent academic environment.

I would like to thank the members of the jury for their beneficial comments and critiques: Bianca Zadrozny, Daniel Schwabe, Fernando Carvalho, Raúl Renteria and Violeta Quental.

My sincere thanks to Raúl Renteria and Lucio Tinoco, both from FAST, a Microsoft Subsidiary, for their encouragement and support.

I am thankful to my colleagues of the LEARN laboratory, Julio Duarte, Roberto Cavalcante, Leandro Alvim, Carlos Crestana, Eraldo Rezende, Breno Faria and Frederico Pessoa. They all have helped me in some way during my PhD study.

Finally, I would like to express my profound gratitude to my wife, mother, sisters and brothers for their moral support and patience during my study in PUC–Rio.

Abstract

Santos, Cícero Nogueira dos; Milidiú, Ruy Luiz. **Entropy Guided Transformation Learning**. Rio de Janeiro, 2009. 85p. D.Sc. Thesis — Department of Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This work presents Entropy Guided Transformation Learning (ETL), a new machine learning algorithm for classification tasks. ETL generalizes Transformation Based Learning (TBL) by automatically solving the TBL bottleneck: the construction of good template sets. ETL uses the information gain in order to select the feature combinations that provide good template sets. ETL provides an effective way to handle high dimensional features. It also enables the inclusion of the *current classification* feature in the generated templates. In this work, we also present ETL Committee, an ensemble method that uses ETL as the base learner. We also show that ETL can use the template evolution strategy to accelerate transformation learning.

We describe the application of ETL to four language independent Natural Language Processing tasks: part-of-speech tagging, phrase chunking, named entity recognition and semantic role labeling. Overall, we apply it to thirteen different corpora in six different languages: Dutch, English, German, Hindi, Portuguese and Spanish.

For each one of the tasks, the ETL modeling phase is quick and simple. It only requires the training set and a simple initial classifier. Our extensive experimental results demonstrate that ETL is an effective way to learn accurate transformation rules. Using a common parameter setting, ETL shows better results than TBL with handcrafted templates for the four tasks. Moreover, the template evolution strategy provides a significant training time reduction for all corpora. Our experimental results also show that ETL Committee improves the effectiveness of ETL classifiers. Using the ETL Committee approach, we obtain state-of-the-art competitive performance results in the thirteen corpus-driven tasks. We believe that by avoiding the use of handcrafted templates, ETL enables the use of transformation rules to a greater range of classification tasks.

Keywords

Entropy guided transformation learning. Transformation based learning. Part-of-speech tagging. Phrase chunking. Named entity recognition. Semantic role labeling.

Resumo

Santos, Cícero Nogueira dos; Milidiú, Ruy Luiz. **Aprendizado de Transformações Guiado por Entropia**. Rio de Janeiro, 2009. 85p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Neste trabalho apresentamos o algoritmo Aprendizado de Transformações Guiado por Entropia (ETL), uma nova estratégia de aprendizado de máquina para tarefas de classificação. ETL generaliza o algoritmo Aprendizado Baseado em Transformações (TBL) resolvendo o gargalo do TBL: a construção de bons gabaritos de regras. ETL usa a métrica ganho de informação para selecionar as combinações de atributos que produzem bons gabaritos. ETL possui um modo efetivo para a manipulação de atributos de alta dimensão. Além disso, permite a inclusão do atributo classificação atual nos gabaritos gerados. Neste trabalho, apresentamos também ETL Committee, uma estratégia para a criação de comitês de classificadores ETL. Descrevemos a aplicação do ETL para quatro tarefas de Processamento de Linguagem Natural para diferentes línguas: etiquetagem morfossintática, identificação de sintagmas, reconhecimento de entidades nomeadas e etiquetagem de papéis semânticos. Em suma, aplicamos o ETL para treze corpora em seis línguas diferentes: holandês, inglês, alemão, hindi, português e espanhol. Para as quatro tarefas, a modelagem com ETL é rápida e simples. Os requisitos são apenas um conjunto de treinamento e um classificador inicial básico. Nossos resultados experimentais extensivos demonstram que o ETL é um meio efetivo para o aprendizado de regras de transformação eficazes. Usando uma configuração comum para as quatro tarefas, o ETL apresenta resultados melhores do que TBL com gabaritos gerados manualmente. Além disso, a estratégia de evolução de gabaritos provê uma significativa redução do tempo de treinamento em todos os casos. Nossos resultados experimentais também mostram que o ETL Committee melhora a eficácia dos classificadores ETL, e obtém resultados competitivos com o estado da arte para os treze corpora examinados. Acreditamos que, ao evitar a necessidade da construção manual de gabaritos, o ETL habilita o uso de regras de transformação para um maior número de tarefas de classificação.

Palavras-chave

Aprendizado de transformações guiado por entropia. Aprendizado baseado em transformações. Etiquetagem morfossintática. Identificação de sintagmas. Reconhecimento de entidades nomeadas. Etiquetagem de papéis semânticos.

Contents

1	Introduction	12
2	Entropy Guided Transformation Learning	16
2.1	Transformation Based Learning	16
2.2	TBL Bottleneck	18
2.3	Entropy Guided Template Generation	20
	<i>Information Gain</i>	20
	<i>Decision Trees</i>	21
	<i>Template Extraction</i>	21
	<i>True Class Trick</i>	22
	<i>High Dimensional Features</i>	23
2.4	Template Evolution	24
2.5	Template Sampling	25
2.6	Redundant Transformation Rules	25
2.7	Related Work	26
3	ETL Committee	28
3.1	Training Phase	29
	<i>Bootstrap Sampling</i>	29
	<i>Feature Sampling</i>	30
	<i>ETL Training</i>	31
3.2	Classification Phase	31
3.3	Related Work	31
4	General ETL Modeling for NLP Tasks	33
4.1	Modeling	33
4.2	Basic Parameter Setting	34
4.3	Committee Parameter Setting	35
4.4	Performance Measures	35
4.5	Software and Hardware	36
5	Part-of-Speech Tagging	37
5.1	Corpora	37
5.2	POS Tagging Modeling	38
	<i>Morphological Stage</i>	38
	<i>Contextual Stage</i>	39
5.3	Machine Learning Modeling	39
5.4	Mac-Morpho Corpus	40
5.5	Tycho Brahe Corpus	40
5.6	TIGER Corpus	41

5.7	Brown Corpus	42
5.8	Summary	42
6	Phrase Chunking	44
6.1	Corpora	44
6.2	Phrase Chunking Modeling	45
	<i>Derived Features</i>	45
6.3	Machine Learning Modeling	46
6.4	SNR-CLIC Corpus	47
6.5	Ramshaw and Marcus Corpus	47
6.6	CoNLL-2000 Corpus	48
6.7	SPSAL-2007 Corpus	49
6.8	Summary	50
7	Named Entity Recognition	51
7.1	Corpora	51
7.2	Named Entity Recognition Modeling	52
	<i>Derived Features</i>	53
7.3	Machine Learning Modeling	53
7.4	HAREM Corpus	54
7.5	SPA CoNLL-2002 Corpus	56
7.6	DUT CoNLL-2002 Corpus	57
7.7	Summary	59
8	Semantic Role Labeling	60
8.1	Corpora	60
8.2	Semantic Role Labeling Modeling	62
	<i>Derived Features</i>	62
	<i>Preprocessing</i>	64
	<i>Postprocessing</i>	64
8.3	Machine Learning Modeling	65
8.4	CoNLL-2004 Corpus	66
8.5	CoNLL-2005 Corpus	67
8.6	Summary	70
9	Conclusions	71
	Bibliography	74
A	ETL Committee Behavior	82
A.1	Ensemble Size	82
A.2	Feature Sampling	82
A.3	Redundant Rules	83
A.4	ETL Committee Random Nature	84

List of Figures

2.1	Transformation Based Learning.	18
2.2	Entropy Guided Transformation Learning.	20
2.3	Decision Tree Learning.	22
2.4	Decision Tree template extraction.	22
3.1	ETL Committee training phase.	30
3.2	ETL Committee classification phase.	32
8.1	Example of an annotated sentence from the CoNLL 2004 Corpus.	62
8.2	Example of a sentence in the collapsed tokens version.	64
8.3	Token sequences for the target verbs <i>stew</i> and <i>put</i> .	65
A.1	$F_{\beta=1}$ x Number of committee members curve.	83

List of Tables

1.1	Corpus characteristics.	13
1.2	System performances.	14
2.1	ETL Template Evolution.	25
4.1	Examples derived from a sequence of tokens.	34
5.1	Part-of-Speech Tagging Corpora.	38
5.2	System performances for the Mac-Morpho Corpus.	40
5.3	System performances for the Tycho Brahe Corpus.	41
5.4	System performances for the TIGER Corpus.	41
5.5	System performances for the Brown Corpus.	42
6.1	Phrase Chunking Corpora.	45
6.2	System performances for the SNR-CLIC Corpus.	47
6.3	System performances for the Ramshaw & Marcus Corpus.	48
6.4	System performances for the CoNLL-2000 Corpus.	48
6.5	ETL _{CMT} results by chunk type for the CoNLL-2000 Corpus.	49
6.6	System performances for the SPSAL-2007 Corpus.	49
7.1	Named Entity Recognition Corpora.	52
7.2	System performances in the Total Scenario for the HAREM Corpus.	55
7.3	System performances in the Selective Scenario for the HAREM Corpus.	55
7.4	ETL _{CMT} results by entity type for the HAREM Corpus.	56
7.5	System performances for the SPA CoNLL-2002 Corpus.	57
7.6	ETL _{CMT} results by entity type for the SPA CoNLL-2002 Corpus.	57
7.7	System performances for the DUT CoNLL-2002 Corpus.	58
7.8	ETL _{CMT} results by entity type for the DUT CoNLL-2002 Corpus.	58
8.1	Semantic Role Labeling Corpora.	61
8.2	System performances for the CoNLL-2004 Corpus.	67
8.3	ETL _{CMT} results by argument type for the CoNLL-2004 Corpus.	68
8.4	System performances for the CoNLL-2005 Corpus.	69
8.5	ETL _{CMT} results by argument type for the CoNLL-2005 Corpus.	69
A.1	ETL Committee performance sensitivity to the percentage of sampled features.	83
A.2	Contribution of redundant rules for the ETL Committee performance.	84
A.3	Results of ten different ETL Committee runs.	84

*The greatest challenge to any thinker is stating
the problem in a way that will allow a solution.*

Bertrand Russell

1 Introduction

Since the last decade, Machine Learning (ML) has proven to be a very powerful tool to help in the construction of Natural Language Processing (NLP) systems, which would otherwise require an unfeasible amount of time and human resources. Transformation Based Learning (TBL) is a ML algorithm introduced by Eric Brill [13] to solve NLP tasks. TBL is a corpus-based, error-driven approach that learns a set of ordered transformation rules which correct mistakes of a baseline classifier. It has been used for several important NLP tasks, such as part-of-speech tagging [13, 24, 65], phrase chunking [59, 43, 61], named entity recognition [26, 44] and semantic role labeling [35].

TBL rules must follow patterns, called templates, that are meant to capture the relevant feature combinations. TBL templates are handcrafted by problem experts. Its quality strongly depends on the problem expert skills to build them. Even when a template set is available for a given task, it may not be effective when we change from a language to another. When the number of features to be considered is large, the effort to manually create templates is extremely increased, becoming sometimes infeasible. Hence, the human driven construction of good template sets is a bottleneck on the effective use of the TBL approach.

In this thesis, we present Entropy Guided Transformation Learning (ETL), a new machine learning algorithm for classification tasks. ETL generalizes Transformation Based Learning by automatically solving the TBL bottleneck: the construction of good template sets. ETL uses the information gain in order to select the feature combinations that provide good template sets. ETL provides an effective way to handle high dimensional features. It also enables the inclusion of the *current classification* feature in the generated templates. In this work, we also present ETL Committee, an ensemble method that uses ETL as the base learner. ETL Committee is particularly useful when dealing with very complex tasks that use large sets of features. We also show that ETL can use the *template evolution* strategy [20] to accelerate transformation learning.

We describe the application of ETL to four language independent NLP

tasks: Part-of-Speech (POS) Tagging, Phrase Chunking (PCK), Named Entity Recognition (NER) and Semantic Role Labeling (SRL). POS tagging is the process of assigning a POS or another lexical class marker to each word in a text [37]. PCK consists in dividing a text into non-overlapping phrases [73]. NER is the problem of finding all proper nouns in a text and to classify them among several given categories of interest [44]. SRL is the process of detecting basic event structures such as *who* did *what* to *whom*, *when* and *where* [70]. These tasks have been considered fundamental for more advanced NLP applications [71, 73, 74, 75]. Overall, we apply ETL to thirteen different corpora in six different languages: Dutch, English, German, Hindi, Portuguese and Spanish. Our goal in these experiments is to assess the robustness and predictive power of the ETL strategy. In Table 1.1, we enumerate the thirteen corpora used throughout this work. For each corpus, we indicate its corresponding language, task and size. The reported size is in terms of sentences and tokens. A *token* is a word or a punctuation mark.

Table 1.1: Corpus characteristics.

Task	Corpus	Language	Sentences	Tokens
POS	Mac-Morpho	Portuguese	53,374	1,221,465
	Tycho Brahe	Portuguese	40,932	1,035,592
	Brown	English	57,340	1,161,192
	TIGER	German	50,474	888,578
PCK	SNR-CLIC	Portuguese	4,392	104,144
	Ramshaw & Marcus	English	10,948	259,104
	CoNLL-2000	English	10,948	259,104
	SPSAL-2007	Hindi	1,134	25,000
NER	HAREM	Portuguese	8,142	165,102
	SPA CoNLL-2002	Spanish	9,840	316,248
	DUT CoNLL-2002	Dutch	21,001	271,925
SRL	CoNLL-2004	English	10,607	251,766
	CoNLL-2005	English	42,248	1,006,712

For each one of the tasks, the ETL modeling phase is quick and simple. It only requires the training set and a simple initial classifier. Moreover, we use a common parameter setting for the four tasks. ETL also simplifies the incorporation of new input features, since it does not require handcrafted templates.

In Table 1.2, we summarize, for each corpus, the performance of both ETL and the state-of-the-art system. The performance measure for the POS tagging task is accuracy. For the other three tasks, the performance measure is the $F_{\beta=1}$. In Table 1.2, the best observed results are in bold. Using the ETL approach, we obtain competitive performance results for the thirteen

corpora. For each one of the tasks, ETL shows better results than TBL with handcrafted templates. ETL Committee improves the effectiveness of ETL classifiers in all cases and achieves state-of-the-art results for six corpora. We believe that by avoiding the use of handcrafted templates, ETL enables the use of transformation rules to a greater range of classification tasks.

Table 1.2: System performances.

Task	Corpus	State-of-the-art		ETL	
		System	Performance	Single	Committee
POS	Mac-Morpho	TBL	96.60	96.75	96.94
	Tycho Brahe	TBL	96.63	96.64	96.72
	Brown	TBL	96.67	96.69	96.83
	TIGER	TBL	96.53	96.57	96.68
PCK	SNR-CLIC	TBL	87.71	88.85	89.58
	Ramshaw & Marcus	SVM	94.22	92.80	93.29
	CoNLL-2000	SVM	94.12	92.28	93.27
	SPSAL-2007	HMM + CRF	80.97	78.53	80.44
NER	HAREM	CORTEX	61.57	61.32	63.56
	SPA CoNLL-2002	AdaBoost	79.29	76.28	77.46
	DUT CoNLL-2002	AdaBoost	77.05	74.18	75.44
SRL	CoNLL-2004	SVM	69.49	63.37	67.39
	CoNLL-2005	AdaBoost	75.47	70.08	72.23

The key concepts of both ETL and ETL Committee evolved through a series of previously published papers by This Author. In Santos & Milidiú [66], we detail the ETL algorithm and apply it to language independent POS tagging, PCK and NER. In Milidiú et al. [48], we present ETL models for three Portuguese language processing tasks: POS tagging, PCK and NER. In Santos et al. [65], we detail the application of ETL to Portuguese POS tagging. In Milidiú et al. [49], we show the application of ETL to language independent PCK. In Santos & Milidiú [62], we present the ETL algorithm.

Additionally, some improvements on TBL are described in a series of previously published papers also by This Author. In Milidiú et al. [46, 47], we present an evolutionary scheme based on genetic algorithms to automatically generate TBL templates. In Santos & Milidiú [63], we show a novel method which enables a TBL classifier to generate a probability distribution over the class labels. In Milidiú et al. [45], we combine Hidden Markov Models (HMM) and TBL in a semi-supervised learning approach. In Freitas et al. [33], we show the application of TBL and HMM to the identification of appositives.

This work is organized as follows. In chapter 2, the ETL algorithm is detailed. In chapter 3, we describe an effective method to create ETL committees. In chapter 4, we show the general ETL modeling for NLP tasks.

In chapter 5, we report our findings on the application of ETL to POS tagging. In chapter 6, we detail the application of ETL to the PCK task. In chapter 7, we report our findings on the application of ETL to the NER task. In chapter 8, we detail the application of ETL to the SRL task. Finally, in chapter 9, we present our concluding remarks and future work.

2

Entropy Guided Transformation Learning

Entropy Guided Transformation Learning (ETL) [62, 49, 65, 48, 66] is a new machine learning algorithm for classification tasks. The ETL approach generalizes Transformation Based Learning by automatically solving the TBL bottleneck: the construction of good template sets. In this chapter, we present the ETL algorithm. In section 2.1, we provide an overview of Transformation Based Learning. Section 2.2 presents the TBL bottleneck. In section 2.3, we detail the ETL learning strategy. In sections 2.4, 2.5 and 2.6 we present some variations on the basic ETL strategy. Finally, in section 2.7, we present the related works.

2.1 Transformation Based Learning

Transformation Based Learning is a machine learning algorithm for classification tasks introduced by Eric Brill [13]. TBL is a corpus-based, error-driven approach that learns a set of ordered transformation rules which correct mistakes of a baseline classifier. It has been used for several Natural Language Processing tasks, such as part-of-speech tagging [13, 24, 65], phrase chunking [59, 43, 61, 45], spelling correction [40], appositive extraction [33], named entity recognition [26, 44] and semantic role labeling [35].

The following three rules illustrate the kind of transformation rules used throughout this thesis.

$$\begin{aligned} & \text{pos}[0] = \text{ART} \quad \text{pos}[1] = \text{ART} \rightarrow \text{pos}[0] = \text{PREP} \\ & \text{pos}[0] = \text{ART} \quad \text{pos}[1] = \text{V} \quad \text{word}[0] = \text{a} \quad \rightarrow \text{pos}[0] = \text{PREP} \\ & \text{pos}[0] = \text{N} \quad \text{pos}[-1] = \text{N} \quad \text{pos}[-2] = \text{ART} \rightarrow \text{pos}[0] = \text{ADJ} \end{aligned}$$

These rules were learned for Portuguese POS tagging. They check the following features: `pos[0]`, the POS tag of the current word; `pos[1]`, the POS tag of the next word; `pos[-1]`, the POS tag of the previous word; `pos[-2]`, the POS tag of the word two before; and `word[0]`, the current lexical item. The first rule should be read as

“IF the POS tag of the current word is an *article*
AND the POS tag of the next word is also an *article*
THEN change the POS tag of the current word to *preposition* ”

TBL rules are composed of two parts: the left hand side and the right hand side. The *left hand side* is a conjunction of feature=value tests, whereas the *right hand side* indicates a value assignment to a target feature. TBL rules must follow patterns, called *rule templates*, that specify which feature combinations should appear in the rule left-hand side. The template set defines the candidate rules space to be searched. Briefly, a template is an uninstantiated rule. The following three templates were used to create the previously shown rules.

```

pos [0]  pos [1]
pos [0]  pos [1]  word [0]
pos [0]  pos [-1]  pos [-2]

```

TBL requires three main inputs:

- (i) a correctly labeled training set;
- (ii) an initial classifier, the *Baseline System* (BLS), which provides an initial labeling for the training examples. Usually, the BLS is based on simple statistics of the correctly labeled training set, such as to apply the most frequent class;
- (iii) a set of rule templates.

The TBL algorithm is illustrated in Figure 2.1. The central idea in the TBL learning process is to greedily learn rules that incrementally reduces the number of classification errors produced by the Initial Classifier. At each iteration, the algorithm learns the rule that has the highest *score*. The score of a rule r is the difference between the number of errors that r repairs and the number of errors that r creates. The learning process stops when there are no more rules whose score is above a given threshold. The *rule score threshold* is a parameter of TBL. A pseudo-code of TBL is presented in Algorithm 1. In this pseudo-code, the *apply* function classifies the given training set examples using the given initial classifier or transformation rule. The *isWronglyClassified* function checks whether the example is misclassified or not. This checking is done by comparing the current example class to the correct class. The *instantiateRule* function creates a new rule by instantiating the given template with the given example context values. The *countCorrections* function returns the

number of corrections that a given rule would produce in the current training set. Similarly, the *countErrors* function returns the number of misclassifications that a given rule would produce in the current training set. There are also several variants of the TBL algorithm. FastTBL [25] is the most successful, since it achieves a significant speedup in the training time while still achieving the same performance as the standard TBL algorithm. We have also developed a TBL variant that produces probabilistic classifications [63].

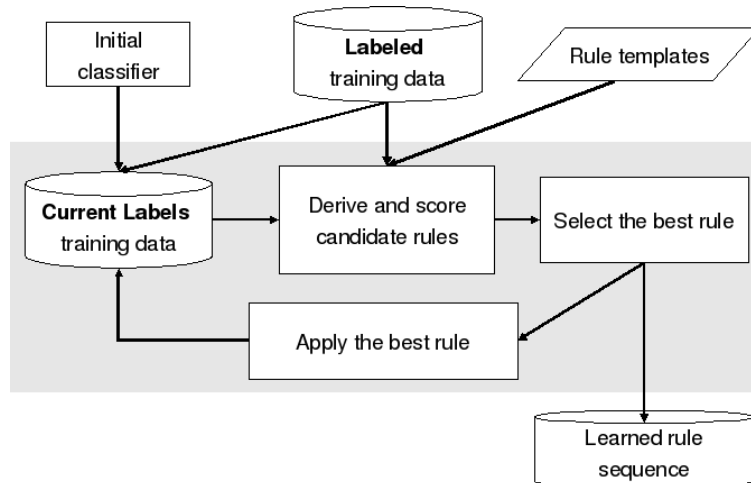


Figure 2.1: Transformation Based Learning.

When using a TBL rule set to classify new data, we first apply the Initial Classifier to the new data. Then we apply the learned rule sequence. The rules must be applied following the same order they were learned.

2.2 TBL Bottleneck

TBL templates are meant to capture relevant feature combinations. Templates are handcrafted by problem experts. Therefore, TBL templates are task specific and their quality strongly depends on the problem expert skills to build them. For instance, Ramshaw & Marcus [59] propose a set of 100 templates for the phrase chunking task. Radu Florian [26] proposes a set of 133 templates for the named entity recognition task. Higgins [35] handcrafted 130 templates to solve the semantic role labeling task. Elming [23] handcrafted 70 templates when applying TBL for machine translation.

The development of effective template sets is a difficult task. It usually involves a lengthy trial-and-error strategy or the adaptation of an existent, known-to-perform-well-on-a-similar-task template set to the new task [27]. The template developer should indicate the relevant feature combinations, otherwise the TBL algorithm can not learn effective rules. When the number of features to be considered is large, the effort to manually combine them

Algorithm 1 Transformation Based Learning Pseudo-Code

```

input LabeledTrainingSet; TemplateSet;
       InitialClassifier; RuleScoreThreshold
1: LearnedRules  $\leftarrow$  {}
2: CurrentTrainingSet  $\leftarrow$  apply(InitialClassifier, LabeledTrainingSet)
3: repeat
4:   CandidateRules  $\leftarrow$  {}
5:   for all example  $\in$  CurrentTrainingSet do
6:     if isWronglyClassified(example) then
7:       for all template  $\in$  TemplateSet do
8:         rule  $\leftarrow$  instantiateRule(template, example)
9:         CandidateRules  $\leftarrow$  CandidateRules + rule
10:      end for
11:    end if
12:  end for
13:  bestScore  $\leftarrow$  0
14:  bestRule  $\leftarrow$  Null
15:  for all rule  $\in$  CandidateRules do
16:    good  $\leftarrow$  countCorrections(rule, CurrentTrainingSet)
17:    bad  $\leftarrow$  countErrors(rule, CurrentTrainingSet)
18:    score  $\leftarrow$  good - bad
19:    if score > bestScore then
20:      bestScore  $\leftarrow$  score
21:      bestRule  $\leftarrow$  rule
22:    end if
23:  end for
24:  if bestScore > RuleScoreThreshold then
25:    CurrentTrainingSet  $\leftarrow$  apply(bestRule, CurrentTrainingSet)
26:    LearnedRules  $\leftarrow$  LearnedRules + bestRule
27:  end if
28: until bestScore > RuleScoreThreshold
output LearnedRules

```

is extremely increased, since there are $2^{|\mathcal{F}|}$ feature combinations, where \mathcal{F} denotes the feature space. On the other hand, the template developer can not generate a very large number of templates, since the training time and memory requirements become intractable. Moreover, Ramshaw & Marcus [58] argue that overfitting is likely when irrelevant templates are included. *Overfitting* is the phenomenon of training too complex a model that do not generalize for new data.

Even when a template set is available for a given task, it may not be effective when we change from a language to another. For instance, Santos & Oliveira [61] extend the Ramshaw & Marcus [59] template set, which was handcrafted for English phrase chunking, by adding six templates specifically designed for Portuguese phrase chunking.

Based on the above, it can be concluded that the human driven construction of good template sets is a bottleneck on the effective use of the TBL approach.

2.3 Entropy Guided Template Generation

Entropy Guided Transformation Learning uses Information Gain (IG) in order to select the feature combinations that provide good template sets. ETL provides an effective way to handle high dimensional features. It also enables the inclusion of the *current classification* feature in the generated templates. The ETL algorithm is illustrated in the Figure 2.2.

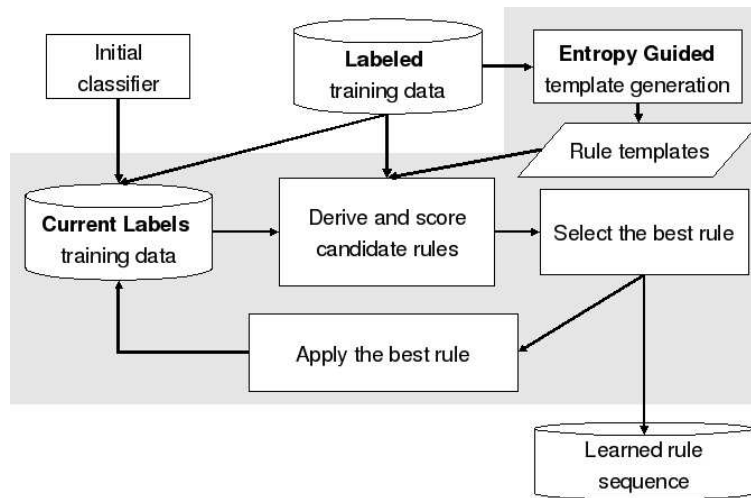


Figure 2.2: Entropy Guided Transformation Learning.

Information Gain (IG), which is based on the data entropy, is a key measure for many feature selection strategies. The most popular Decision Tree (DT) learning algorithms [57, 69] use this measure. Hence, they provide a quick way to obtain entropy guided feature selection. In the ETL strategy, we use DT induction algorithms to automatically generate template sets.

The remainder of this section is organized as follows. First, we review the IG measure and the DT learning method. Next, we show how to automatically generate templates from decision trees. Next, we present the ETL true class trick. Finally, we detail how ETL handles high dimensional features.

(a) Information Gain

Information gain is a statistical measure commonly used to assess feature relevance [21, 29, 56]. IG is based on the Entropy concept, which characterizes the impurity of an arbitrary collection of examples. Given a training set T

whose examples assume classes from the set C . The entropy of T relative to this classification is defined as

$$H(T) = - \sum_{i=1}^{|C|} P_T(c_i) \log_2 P_T(c_i) \quad (1)$$

where c_i is a class label from C , $|C|$ is the number of classes and $P_T(c_i)$ is estimated by the percentage of examples belonging to c_i in T .

In feature selection, information gain can be thought as the expected reduction in entropy $H(T)$ caused by using a given feature A to partition the training examples in T . The information gain $IG(T, A)$ of a feature A , relative to an example set T is defined as

$$IG(T, A) = H(T) - \sum_{v \in \text{Values}(A)} \frac{|T_v|}{|T|} H(T_v) \quad (2)$$

where $\text{Values}(A)$ is the set of all possible values for feature A , and T_v is the subset of T for which feature A has value v [50]. When using information gain for feature selection, a feature A is preferred to feature B if the information gain from A is greater than that from B .

(b) Decision Trees

Decision tree induction is a widely used machine learning algorithm [56]. Quinlan's C4.5 [57] system is the most popular DT induction implementation. It recursively partitions the training set using the feature providing the largest information gain. This results into a tree structure, where the nodes correspond to the selected features and the arc labels to the selected feature values. After the tree is grown, a pruning step is carried out in order to avoid overfitting.

In Figure 2.3, we illustrate the DT induction process for Portuguese POS tagging. Here, the five selected features are: `pos[0]`, the POS tag of the current word; `pos[-1]`, the POS tag of the previous word; `pos[1]`, the POS tag of the next word; `pos[-2]`, the POS tag of the word two before; and `word[-1]`, the previous lexical item. The feature values are shown in the figure as arc labels.

We use the C4.5 system to obtain the required entropy guided selected features. We use pruned trees in all experiments shown here.

(c) Template Extraction

In a DT, the more informative features appear closer to the root. Since we just want to generate the most promising templates, we combine first the more informative features. Hence, as we traverse the DT from the root to a leaf, we collect the features in this path. This feature combination provides an

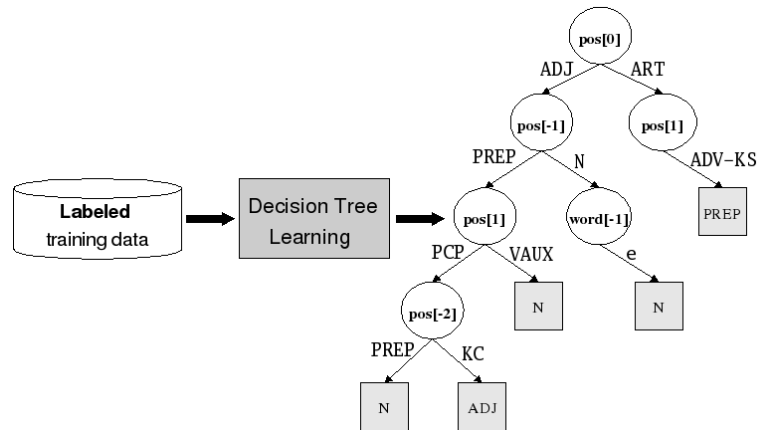


Figure 2.3: Decision Tree Learning.

information gain driven template. Additionally, paths from the root to internal nodes also provide good templates.

It is very simple to obtain these templates from C4.5's output. From the given DT, we eliminate the leaves and the arc labels. We keep only the tree structure and the node labels. Next, we execute a depth-first traversal of the DT. For each visited tree node, we create a template that combines the features in the path from the root to this node. Figure 2.4 illustrates the template extraction process. In this figure, the template in bold is extracted from the tree path in bold.

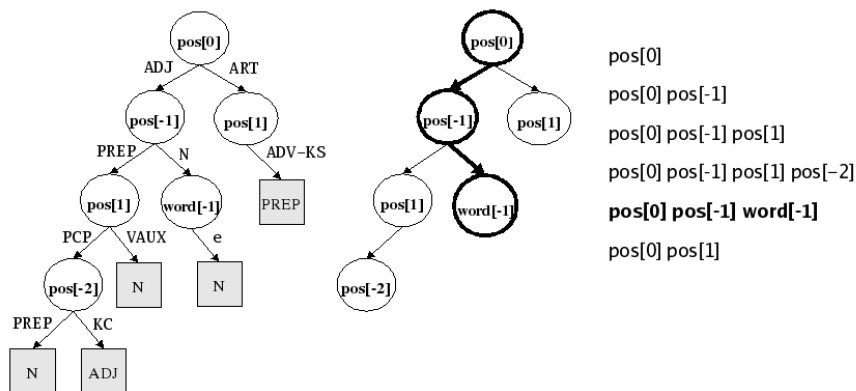


Figure 2.4: Decision Tree template extraction.

(d) True Class Trick

TBL learning can access intermediate results of the classification process as a feature. Moreover, due to the iterative nature of the error correction approach adopted by TBL, the information in the *current classification feature* becomes more precise throughout the rule application process. For instance, when applying TBL to NLP tasks, one usual feature is the current classification

of the words within a context window. This behavior is very desirable for NLP tasks, where local classification dependencies play an important role. In order to explore this TBL property in the ETL approach, the current classification feature must be available for selection in the template generation process. We include this feature by providing the DT learner with the initial and true classification of the words. We call this strategy as the *true class trick*.

When creating DT examples, we use the following information as the value of the current classification features: (1) the initial class label for the target word; and (2) the true class label for the neighbor words. Using the true class labels for the neighbor words produce better results, since they contain a precise information. This reflects what the TBL algorithm finds in the context window after some iterations. At this point, the total number of remaining errors is small, dispersed and spread throughout the data. Hence, around an incorrectly classified word is very likely that all the words are correctly classified. Using the true class labels for the target word is not allowed, since it would imply the use of the task solution as an input feature.

The use of the true class labels at training time is a general modeling strategy. It is usual for algorithms that do not have access to intermediate classification results, such as DT and Support Vector Machines (SVM). For instance, when training an AdaBoost system for SRL, Surdeanu et al. [71] use the true class labels of the left side words. At test time, they use the class labels predicted by the classifier, since the classification is done in a left to right fashion. Kudo & Matsumoto [38] use a similar strategy when applying SVM for Phrase Chunking. The advantage of TBL over these algorithms is that, by default, it has access to the current classification of the words in both left and right sides, both at training and at test time. Furthermore, the ETL *true class trick* allows the effective use of this TBL property.

(e) High Dimensional Features

High dimensional features are characterized by having a large number of possible values. For instance, the feature *words* in a text is high dimensional, since it can assume thousands of different values. This kind of feature is ineffective when information gain is used as the feature informativeness measure. IG has a bias that favors high dimensional features over those with a few values [50]. Since we use information gain in the ETL method, through DT learning, we must overcome this problem.

Although the C4.5 system uses the IG ratio measure to avoid the IG bias, the use of high dimensional features is still ineffective. When these features are present in the training set, usually the C4.5 system does not make use of them.

Therefore, we include a preprocessing step. This step consists in pruning the high dimensional features in order to retain only their most informative values.

Let T be a training set that contains the high dimensional feature A . For each value v that A assumes in T , we compute its individual information gain using the following equation.

$$IG(T, A, v) = H(T) - \frac{|T_v|}{|T|} H(T_v) \quad (3)$$

where $H(T)$ is the entropy of the training set T , T_v is the subset of T for which feature A has value v , and $H(T_v)$ is the entropy of the subset T_v . After the individual IGs are computed, we sort the feature values in decreasing order of IG. Let SV be the set containing the top z IG values. Then, for each example in T whose value for A is not in SV , we replace that value by a common dummy value. Observe that z is a parameter of the ETL algorithm.

Note that this preprocessing is used only at the DT learning stage. All feature values are maintained at the transformation rule learning stage.

2.4 Template Evolution

TBL training time is highly sensitive to the number and complexity of the applied templates. Curran & Wong [20] argued that we can better tune the *training time* vs. *templates complexity* trade-off by using an evolutionary template approach. The main idea is to apply only a small number of templates that evolve throughout the training. When training starts, templates are short, consisting of few feature combinations. As training proceeds, templates evolve to more complex ones that contain more feature combinations. In this way, only a few templates are considered at any point in time. Nevertheless, the descriptive power is not significantly reduced.

ETL provides an easy scheme to implement the template evolution strategy. First, we partition the learned template set by template size. Let T_k be the template set containing all templates of size k , where $k = 1, \dots, K$ and K equals to the largest template size. Next, we split the TBL step into K consecutive phases. In phase k , TBL learns rules using only templates from T_k . For instance, using the tree shown in Figure 2.4, we have four TBL training phases. In Table 2.1, we show the template sets used in the four TBL phases when the tree shown in Figure 2.4 is used.

Using the template evolution strategy, the training time is decreased by a factor of five for the English phrase chunking task. This is a remarkable reduction, since we use an implementation of the *fastTBL* algorithm [51] that is already a very fast TBL version. Training time is a very important issue when modeling a system with a corpus-based approach. A fast ML strategy

Table 2.1: ETL Template Evolution.

Phase	Template set
1	pos[0]
2	pos[0] pos[-1] pos[0] pos[1]
3	pos[0] pos[-1] pos[1] pos[0] pos[-1] word[-1]
4	pos[0] pos[-1] pos[1] pos[-2]

enables the testing of different modeling options, such as different feature sets. The efficacy of the rules generated by ETL template evolution is quite similar to the one obtained by training with all the templates at the same time.

2.5 Template Sampling

Although the template evolution strategy produces a significant speedup in the ETL training time, it has a poor performance when it is necessary to learn all the possible rules at each consecutive TBL phase. If an excessive number of rules are learned in the earlier phases, the posterior phases will have just a few errors to correct. However, the main problem is that the templates of the earlier phases are very simple and may generate poor rules if all the rules with positive score are learned. Therefore, in cases where it is necessary to learn the largest rule set possible, template evolution is not suitable. On the other hand, in many cases this is not a problem. Usually, in order to avoid overfitting, we only learn rules whose score is at least two.

Nevertheless, there are cases where the learning of the largest rule set is necessary. For instance, when training an ensemble of classifiers using different training data sets, overfitting can be beneficial. This is because, in this specific case, overfitting can introduce diversity among the ensemble members. As an example, some DT ensemble learning methods do not use pruning [5, 12, 36].

In our ETL implementation, we also include the *template sampling* functionality, which consists in training the ETL model using only a randomly chosen fraction of the generated templates. Besides being simple, this strategy provides a speed up control that is very useful when multiple ETL models are to be learned.

2.6 Redundant Transformation Rules

As previously noticed by Florian [27], the TBL learning strategy shows a total lack of redundancy in modeling the training data. Only the rule that

corrects the largest number of errors is selected at each learning iteration. All alternative rules that may correct the same errors, or a subset of the errors, are ignored. This greedy behavior is not a problem when the feature values tested in the alternative rules and the ones tested in the selected rule always co-occur. Unfortunately, this is not always the case when dealing with sparse data.

Florian includes redundancy in his TBL implementation by adding to the list of rules, after the training phase has completed, all the rules that do not introduce error. Florian shows that these additional rules improve the TBL performance for tasks where a word classification is independent of the surrounding word classifications.

In our ETL implementation, we also include redundancy in the TBL step, but in a different way. At each iteration, when the best rule b is learned, the algorithm also learns all the rules that do not include errors and correct exactly the same examples corrected by b . These redundant rules do not alter the error-driven learning strategy, since they do not provide any change in the training data. Their inclusion is compatible with the standard TBL framework, in the sense that applying the resulting rule set for the training data results in the same number of errors, with or without redundant rules. This kind of redundancy is more effective for low scored rules, since they are more likely to use sparse feature values and their selection is supported by just a few examples.

For the four tasks presented in this work, the inclusion of redundant rules does not improve the performance of single ETL classifiers. Actually, in some cases there is a decrease in the classification performance. We believe this performance degradation is due to a greater overfitting. Redundant rules increase the overfitting because more information from the training set is included in the learned model. However, the inclusion of redundancy improves the classification quality when several classifiers are combined. Since overfitting can be beneficial when multiple classifiers are used.

2.7 Related Work

Liu et al. [39] present a method for automatic template generation that uses DT. In their method, the Tree-guided Transformation-based Learning (TTBL), two DT's are generated: one that uses all the examples and another that uses only the examples wrongly classified by the Initial Classifier. They produce the template set by extracting templates from the two trees. They use the second tree with the aim of producing templates that focus on the errors in the initial classification. Liu et al. apply the TTBL strategy to eliminate

homograph ambiguity in a Mandarin text-to-speech system. For this task, TTBL obtains results comparable to the ones of handcrafted templates. TTBL is similar to ETL in the sense that they extract templates from DTs. However, the ETL template generation process is more versatile, since it uses the initial classification as an input feature. The ETL *true class trick* enables the use of the current classification feature in an effective way. Moreover, ETL [62] has been published earlier than TTBL [39].

An evolutionary scheme based on Genetic Algorithms (GA) to automatically generate TBL templates is presented by Milidiú et al. [46, 47]. Using a simple genetic coding, the generated template sets show an efficacy close to the one of handcrafted templates. The main drawback of this strategy is that the GA step is computationally expensive, since it is necessary to run the TBL algorithm in order to compute the fitness for each individual of the population. If we need to consider a large context window or a large number of features, it becomes infeasible.

Corston-Oliver & Gamon [19] present a combination of DTs and TBL. They derive candidate rules from the DT, and use TBL to select and apply them. Their work is restricted to binary features only. ETL strategy extracts more general knowledge from the DT, since it builds rule templates. Furthermore, ETL is applied to any kind of discrete features.

Carberry et al. [15] introduce a randomized version of the TBL framework. For each error, they try just a few randomly chosen templates from the given template set. This strategy speeds up the TBL training process, enabling the use of large template sets. However, they use handcrafted templates and variations of them, which implies that a template designer is still necessary.

Hwang et al [77] use DT decomposition to extract complex feature combinations for the Weighted Probabilistic Sum Model (WPSM). They also extract feature combinations that use only some nodes in a tree path. This is required to improve the effectiveness of the WPSM learning. Their work is similar to ours, since they use DT's for feature extraction. Nevertheless, the ETL learned template set is simpler than theirs, and is enough for effective TBL learning.

3

ETL Committee

The combination of multiple classifiers is a general Machine Learning approach [11, 31, 36, 22, 12, 28]. Ensemble methods are learning algorithms that generate multiple individual classifiers and combine them to classify new samples. Usually, the final classification is done by taking a weighted or majority vote of the individual predictions. Such combinations of models are known as *ensemble models* or *committees*. The main purpose of model combination is to reduce the generalisation error of a classifier. The *generalisation error* is the error that a classifier makes on data not used in the training stage. Ensemble algorithms have received considerable attention in the last years [14, 54, 6, 52, 8, 55].

According to Dietterich [22], a necessary and sufficient condition for an ensemble of classifiers to have a lower generalisation error than any of its individual members is that the classifiers are accurate and diverse. A classifier is considered to be accurate if its error rate on new data is lower than just guessing. Two classifiers are diverse if they make different errors on new data. Several methods have been suggested for the creation of ensemble of classifiers [22]. Most ensemble methods differ on how they create diversity among the committee members. The manipulation of the training examples is commonly used to provide diverse classifiers [11, 36, 12].

Some general ensemble modeling strategies are Bootstrap Aggregating (Bagging), Random Subspaces and Boosting. The *Bagging* method, introduced by Breiman [11], creates an ensemble of classifiers by making multiple replicates of the training set and using these as new training sets. These replicates are created using bootstrap sampling, which is the process of sampling at random with replacement from a data set. The *Random Subspaces* method, introduced by Ho [36], generates different classifiers by using different randomly chosen feature subsets. The *Adaboost* algorithm, introduced by Freund and Schapire [31], sequentially creates classifiers using a training set with weighted examples. In the Adaboost learning process, examples that are incorrectly predicted by a classifier have their weights increased for the next iteration.

In this chapter, we present ETL Committee, an ensemble method that

uses ETL as a base learner. The ETL Committee strategy relies on the use of training data manipulation to create an ensemble of ETL classifiers. ETL Committee combines the main ideas of Bagging and Random Subspaces. From Bagging, we borrow the bootstrap sampling method. From Random Subspaces, we use the feature sampling idea. In the ETL Committee training, we use ETL with template sampling, which provides an additional randomization step. As far as we know, this is the first study that uses transformation rule learning as the base learner for an ensemble method.

This chapter is organized as follows. In section 3.1, we detail the ETL Committee training phase. In section 3.2, we detail the classification phase. Finally, in section 3.3, we present some related works.

3.1 Training Phase

Given a labeled training set \mathcal{T} , the ETL Committee algorithm generates l ETL classifiers using different versions of \mathcal{T} . In Figure 3.1, we detail the ETL Committee training phase. The creation of each classifier is independent from the others. Therefore, the committee training process can be easily parallelized. In the creation of a classifier c , the first step consists in using *bootstrap sampling* to produce a bootstrap replicate \mathcal{T}' of the training set \mathcal{T} . Next, *feature sampling* is applied to \mathcal{T}' , generating the training set \mathcal{T}'' . Finally, in the *ETL training* step, a rule set is learned using \mathcal{T}'' as a training set. These steps are detailed in the following subsections. In Appendix A, we show some experimental results that highlight the contribution of each one of these steps to the committee behavior.

(a) Bootstrap Sampling

In the *bootstrap sampling* step, a new version of the training set is generated using bootstrapping. *Bootstrapping* consists of sampling at random with replacement from the training set to generate an artificial training set of the same size as the original one. Hence, given a training set \mathcal{T} consisting of n examples, a bootstrap replicate \mathcal{T}' is constructed by sampling n examples at random, with replacement, from \mathcal{T} . Bootstrapping is the central idea of the Bagging ensemble method, where it is used to provide diversity among the ensemble members.

According to Breiman [11], an ensemble of classifiers trained on different bootstrap replicates can be effective if the base learner is *unstable*. An unstable classifier is one where small changes in the training set result in large changes in its predictions. Due to the greedy nature of the TBL learning process, rule

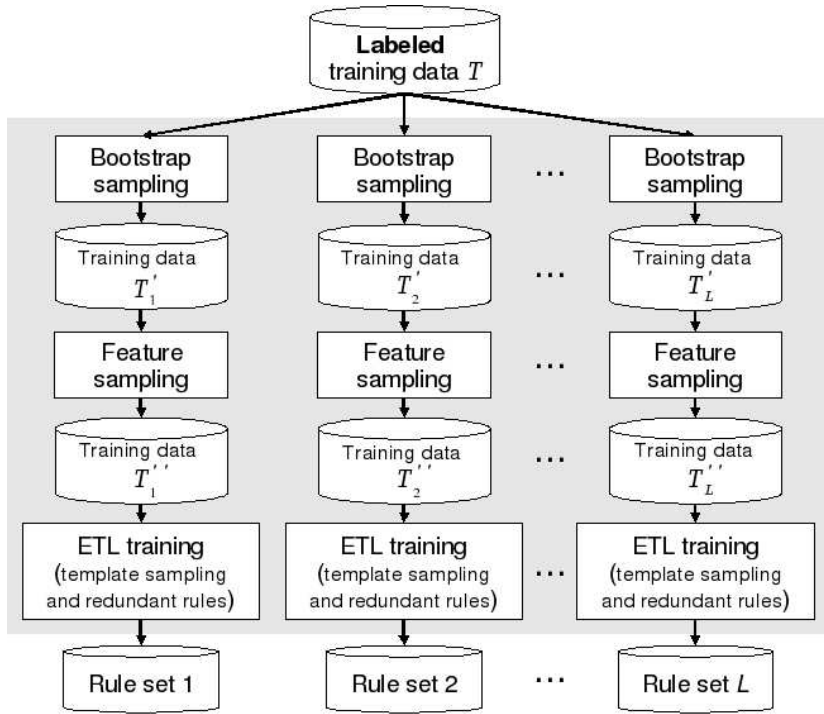


Figure 3.1: ETL Committee training phase.

selection is very sensitive to the occurrence of just a few examples. Usually, the rules in the tail of the learned rule set are selected based on just one or two error corrections. Therefore, we believe that small changes in the training set are able to significantly change the learned rule set. Moreover, since ETL uses DT to obtain templates and DT is an unstable learner [11], there is variability between the template sets generated from different bootstrap replicates. The use of different template sets has the potential to increase the ensemble diversity.

The number of bootstrap replicates is called the *ensemble size*. In Appendix A.1, we investigate the behavior of the ETL Committee performance as the number of committee members increases.

(b) Feature Sampling

In the *feature sampling* step, a new version of the training set is generated by randomly selecting a subset of the available features. The manipulation of the input feature set is a general technique for generating multiple classifiers. As each classifier is generated using a randomly drawn feature subset, the diversity among the ensemble members tends to increase. Feature sampling is the main idea used in the Random Subspaces ensemble method. This strategy is particularly useful when a large set of features is available. The percentage of input features to be included in the subset is a parameter of the ETL Committee method. In Appendix A.2, we analyse the ensemble performance

sensitivity to the percentage of sampled features.

(c) ETL Training

In the *ETL training* step, a set of transformation rules is learned using the training set resulted from the two previous steps. Here, *template sampling* and *redundant transformation rules* are used. We use template sampling for two reasons: (1) it provides more diversity among the ensemble members, since it increases the chance of each classifier to be trained with a very different template set; (2) it speeds up the training process, since less templates are used, enabling the learning of larger rule sets in a reasonable time. Note that by sampling templates we are sampling feature combinations. Hence, the template sampling can be seen as a kind of feature sampling at the base learner level. The number of templates to be sampled is a parameter of the ETL Committee method.

We use redundant rules since it increases the overfitting, and more information from the training set is included in the learned model. Overfitting is another way to introduce diversity among the ensemble members. For instance, DT's are usually not pruned when used as a base learner [36, 12, 5]. We use redundant rules in all ETL Committee experiments reported in this thesis. In Appendix A.3, we analyse the contribution of redundant rules for the ETL Committee performance.

3.2 Classification Phase

In Figure 3.2, we detail the ETL Committee classification phase. When classifying new data, each rule set is independently applied to the input data. For each data point, each ETL model gives a classification, and we say the model “votes” for that class. The final data point classification is computed by majority voting.

A drawback of our proposed ETL Committee, as well as the other ensemble methods, is that it increases the classification time. For instance, when using an ensemble of size 100, the classification time is 100 times slower than the one of a single ETL classifier. However, this process can be easily parallelized, since the application of each rule set is independent from the others.

3.3 Related Work

Breiman [12] presents an ensemble model called *Random Forest*, which uses bootstrapping and feature sampling. In the Random Forest learning

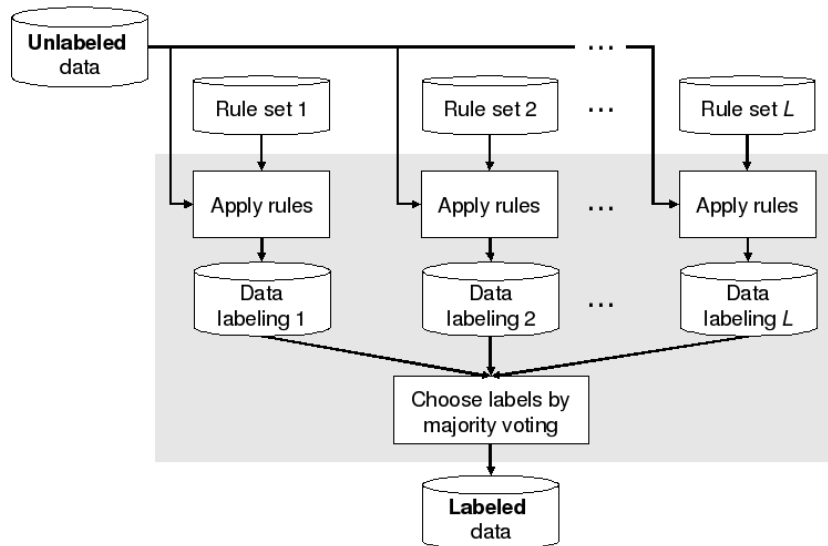


Figure 3.2: ETL Committee classification phase.

process, first, bootstrap sampling is employed to generate multiple replicates of the training set. Then, a decision tree is grown for each training set replicate. When growing a tree, a subset of the available features is randomly selected at each node, the best split available within those features is selected for that node. Each tree is grown to the largest extent possible, and there is no pruning. Random Forest is specific for decision trees, since the feature sampling step occurs at the base learner level. ETL Committee differs from Random Forest in three main aspects: the base learner, where ETL is used; the feature sampling, which is done outside of the base learner; and the template sampling, which is a feature combination sampling method employed at the base learner level.

Panov & Dzeroski [54] describe an ensemble method that also combines Bagging and Random Subspaces. Their intention is to achieve an algorithm whose behaviour is similar to the one of Random Forests, but with the advantage of being applicable to any base learner. Their method uses bootstrap sampling followed by feature sampling to generate different training sets. They show that, when using DT as a base learner, their approach has a comparable performance to that of random forests. The ETL Committee method is similar to the one of Panov & Dzeroski in terms of training set manipulation. On the other hand, ETL Committee differs from the Panov & Dzeroski approach because it includes template sampling, which is a randomization at the base learner level.

4

General ETL Modeling for NLP Tasks

In this chapter we present some general ETL configurations used for NLP tasks. These configurations are the same for all examined tasks. Hence, the ETL modeling phase is performed with little effort. In section 4.1, we show how to model NLP tasks as classification tasks. In section 4.2, we present the basic ETL parameter setting. In section 4.3, we present the ETL Committee parameter setting. In section 4.4, we detail the performance measures used to assess the system performances on NLP tasks. Finally, in section 4.5, we describe the software and hardware used in the experiments.

4.1 Modeling

Although transformation rules can be used in more general processing tasks, ETL is concerned only with classification tasks. Hence, the problems where we apply ETL must be cast as classification problems. Fortunately, most NLP tasks are easily modeled this way.

In this work, the four examined tasks are modeled as token classification problems. Which means that, given a text, the learned system must predict a class label for each text token. Here, a *token* is a word or a punctuation mark. Each token provides an *example*, which corresponds to the list of feature values in the local context of the token. The local context of a token comprises the token itself and the adjacent ones. The number of tokens in the context defines the size of what is called the *context window*. For instance, using a size three context window, the following sentence produces the nine examples shown in Table 4.1. In this case, each token contains the features *word* and *POS tag*. Here, EOS tag indicates the start or end of a sentence.

```
The/DT carriers/NNS were/VBD competing/VBG  
fiercely/RB for/IN market/NN share/NN ./.
```

Table 4.1: Examples derived from a sequence of tokens.

ID	word[-1]	pos[-1]	word[0]	pos[0]	word[1]	pos[1]
1	EOS	EOS	The	DT	carriers	NNS
2	The	DT	carriers	NNS	were	VBD
3	carriers	NNS	were	VBD	competing	VBG
4	were	VBD	competing	VBG	fiercely	RB
5	competing	VBG	fiercely	RB	for	IN
6	fiercely	RB	for	IN	market	NN
7	for	IN	market	NN	share	NN
8	market	NN	share	NN	.	.
9	share	NN	.	.	EOS	EOS

4.2 Basic Parameter Setting

In all experiments presented in this work, we configure ETL as described below. The parameters are empirically tuned using the training and development sets available for the NER and SRL tasks. We also perform a 10-fold cross-validation using the SNR-CLIC Corpus, a Portuguese PCK corpus that is described in Section 6.1. We choose the common parameter setting that provides the best results for these three tasks.

We use this common parameter setting in our experiments with all tasks. This is recommended to verify the robustness of the ETL approach. Next, we list the setup.

Context window: we use a context window of size 7, which is the usual size for the four tasks.

Template size: we use templates which combine at most six features. Therefore, when extracting templates from DTs, the extraction process examines only the six first DT levels.

High dimensional features: we use the high dimensional feature preprocessing step for the feature *word*. We set the z parameter to 200.

Rule score threshold: we let the ETL algorithm learn rules whose score is at least two.

We report results for ETL trained with all the templates at the same time as well as using template evolution. In the template evolution all the parameters are maintained.

4.3 Committee Parameter Setting

In all experiments presented in this work, we configure ETL Committee as described below. We use a common parameter setting in order to simplify our experiments, as well as to verify the robustness of ETL Committee.

Bootstrap Sampling: we use sentences as sampling units for bootstrapping. We set the ensemble size to 100. For all tested data sets, using more than 100 classifiers does not improve significantly the ensemble performance.

Feature Sampling: we use .9 as the default value for the feature sampling parameter. Which means that we randomly choose 90% of the available features when creating each model. Using a greater sampling rate may degrade the results for tasks which have a few number of features. However, we observe that this parameter can be specifically tuned to improve the committee performance for a given training set.

ETL Training: we train the ETL model to learn the largest rule set possible. We use 50 as the default number of templates to be sampled in the creation of each classifier. Using this amount of templates provides a reasonable training time speedup. However, we use 100 templates for SRL. This is because SRL involves a large number of features, which produces a larger number of templates. For instance, almost 650 templates are generated for the SRL task, while about 100 templates are generated for the PCK task.

4.4 Performance Measures

When performing token classification, the *token-by-token accuracy* is the usual system performance measure. The token-by-token accuracy is formalized by the following equation.

$$Accuracy = \frac{\# \text{ of correctly classified tokens}}{\# \text{ of tokens in the test set}} \quad (1)$$

Precision, recall and F-measure are performance measures usually applied for information extraction tasks such as PCK, NER and SRL. *Precision* informs how many correct items the system extracted amongst all extracted items. *Recall* informs how many correct items the system extracted amongst all existing items. *F-measure* is the weighted harmonic mean of precision and recall. As usual, we use the $F_{\beta=1}$ -measure, where recall and precision are evenly weighted. The following equations formalize precision, recall and $F_{\beta=1}$.

$$Precision = \frac{\# \text{ of correctly extracted items}}{\# \text{ of extracted items}} \quad (2)$$

$$Recall = \frac{\# \text{ of correctly extracted items}}{\# \text{ of items in the test set}} \quad (3)$$

$$F_{\beta=1} = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

We also use the $F_{\beta=1}$ error, which is given by the following equation.

$$F_{\beta=1}error = 100 - F_{\beta=1} \quad (5)$$

4.5 Software and Hardware

Our current ETL implementation uses the C4.5 system [57] to learn decision trees. In order to learn transformation rules, we have implemented the fastTBL algorithm [51] using the Python programming language.

The ETL experiments are performed using an Intel[®] Centrino[®] Duo 1.66GHz notebook with 2GB of RAM memory.

The ETL Committee experiments are performed using a cluster of 15 machines. Each machine has 2GB of RAM memory and an Intel[®] Pentium[®] D CPU 3.40GHz with 2 cores. Therefore, the cluster has 30 CPU cores. Since the training of each classifier is independent from the others, the ETL Committee experiments are easily parallelized. When training a committee, we run the same code in the 30 different cores.

5

Part-of-Speech Tagging

Part-of-Speech (POS) tagging is the process of assigning a POS or another lexical class marker to each word in a text [37]. POS tags classify words into categories, based on the role they play in the context in which they appear. The POS tag is a key input feature for NLP tasks like phrase chunking and named entity recognition. In the example that follows, which was extracted from the Mac-Morpho Corpus, the POS tags have the following meanings: N=*noun*, ADJ=*adjective*, KC=*coordinating conjunction*, PREP=*preposition*, and V=*verb*.

```
Safra/N recorde/ADJ e/KC disponibilidade/N de/PREP crédito/N
ativam/V vendas/N de/PREP máquinas/N agrícolas/ADJ ./.
```

This chapter presents the application of the ETL approach to language independent POS tagging. We apply ETL to four different corpora in three different languages. In section 5.1, we describe the selected corpora. In section 5.2, we detail some modeling configurations used in our POS tagger system. In section 5.3, we show some configurations used in the machine learning algorithms. Section 5.4 presents the application of ETL for the Mac-Morpho Corpus. Section 5.5 presents the application of ETL for the Tycho Brahe Corpus. Section 5.6 presents the application of ETL for the TIGER Corpus. Section 5.7 presents the application of ETL for the Brown Corpus. Finally, section 5.8 presents some concluding remarks.

5.1 Corpora

We apply ETL to four different POS tagged corpora in three different languages. The selected corpora are: Mac-Morpho [2], a Portuguese language corpus; Tycho Brahe [72], a Portuguese language corpus; TIGER [10], a German language corpus; and Brown [30], an English language corpus. In Table 5.1, we show some characteristics of these corpora.

The Mac-Morpho Corpus is tagged with 22 POS tags, while the Tycho Brahe Corpus is tagged with 383 POS tags. The TIGER Corpus is tagged with

Table 5.1: Part-of-Speech Tagging Corpora.

Corpus	Language	Tagset size	Training Data		Test Data	
			Sent.	Tokens	Sent.	Tokens
Mac-Morpho	Portuguese	22	44,233	1,007,671	9,141	213,794
Tycho Brahe	Portuguese	383	30,698	775,601	10,234	259,991
TIGER	German	54	41,954	742,189	8,520	146,389
Brown	English	182	47,027	950,975	10,313	210,217

54 POS tags, and the Brown Corpus is tagged with 182 POS tags. Both the Tycho Brahe Corpus and the Brown Corpus use more POS tags because these tags also identify morphological aspects such as word number and gender. Each corpus is divided into training and test sets. For the Portuguese corpora, these training and test set splits are the same as reported by Milidiú et al. [49].

5.2 POS Tagging Modeling

In POS tagging, a word is called a *known word* if it appears in the training set. Otherwise, it is called an *unknown word*. Usually, POS tagging systems handle unknown words in a separate way. Brill [13] proposes a POS tagging modeling for TBL that consists of two stages: the *morphological*, which classifies the unknown words using morphological information; and the *contextual*, which classifies the known and unknown words using contextual information. Our POS tagging modeling approach follows the two stages strategy proposed by Brill. The following subsections detail the two stages.

(a) Morphological Stage

In the morphological stage, *morphological rules* are used to classify the unknown words. TBL is used to learn the morphological rules. These rules are based on the following token features:

- up to c characters long word prefixes and suffixes;
- specific character occurrence in a word;
- adding (or subtracting) a c characters long prefix (or suffix) results in a known word;
- occurrence of the word before (or after) a specific word W in a given long list of word bigrams. For instance, if the word appears after “to”, then it is likely to be a verb in the infinitive form.

In our experiments, we set the parameter c equal to 5.

With a very simple template set [13], one can effectively perform the morphological stage. For this stage, it is enough to use one feature or two feature templates of the token been classified. The one feature templates use one of the current token features. The two feature templates use one of the current token features and the current token POS. We do not use ETL to train the morphological stage, since the templates used here are very simple. Therefore, in our experiments, the TBL-based morphological stage is part of the baseline system.

(b) Contextual Stage

In the contextual stage, contextual rules are used to correct errors in the classification of known and unknown words. The contextual rules use the features *word* and *POS* of any token in the defined context window. We use ETL for learning contextual rules only.

5.3 Machine Learning Modeling

We use the following task specific configurations.

BLS: the baseline system assigns to each word the POS tag that is most frequently associated with that word in the training set. Words that do not appear in the training set are classified by the morphological rules.

TBL: the results for the TBL approach refer to the contextual stage trained using the lexicalized template set proposed by Brill [13]. This template set uses combinations of *words* and *POS* in a context window of size seven.

ETL: in the ETL learning, we use the features *word* and *POS*. The default ETL parametrization shown in Chapter 4 is used.

ETL_{TE}: we use the same ETL configuration, but here we perform template evolution.

ETL_{CMT}: we use the default ETL Committee parametrization shown in Section 4.3.

We also report the state-of-the-art system performance for each corpus.

5.4 Mac-Morpho Corpus

Santos et al. [65] present a TBL system with state-of-the-art performance for the Mac-Morpho Corpus. Therefore, for this corpus, we only report the performance of ETL, TBL and BLS systems.

In Table 5.2, we summarize the system performance results. The ETL system reduces the BLS accuracy error by 61%, from 8.34 to 3.25. ETL and TBL systems achieve similar accuracy. The ETL_{TE} system reduces the ETL training time by 73%, from 214 mins to 58 mins. This is a remarkable reduction, since we use an implementation of the *fastTBL* algorithm [51] that is already a very fast TBL version. Moreover, there is no accuracy reduction when using the template evolution approach. This suggests that, for this task, complex templates are not necessary for the earlier learned rules.

The ETL_{CMT} system reduces the accuracy error by 6% when compared to the single ETL system. Its accuracy, 96.94%, is the best one reported so far for the Mac-Morpho Corpus. On the other hand, the classification with the ETL_{CMT} system is 100 times slower than the one with the single ETL system.

Table 5.2: System performances for the Mac-Morpho Corpus.

System	Accuracy (%)	# Templates
ETL_{CMT}	96.94	50
ETL	96.75	72
ETL_{TE}	96.74	72
TBL	96.60	26
BLS	91.66	–

5.5 Tycho Brahe Corpus

Santos et al. [65] present a TBL system with state-of-the-art performance for the Tycho Brahe Corpus. Therefore, for the Tycho Brahe Corpus, we only report the performance of ETL, TBL and BLS systems.

In Table 5.3, we summarize the system performance results. The ETL system reduces the BLS accuracy error by 52%, from 7.00 to 3.36. ETL and TBL systems achieve similar accuracy. The ETL_{TE} system reduces the ETL training time by 65%, from 60 mins to 21 mins. Moreover, there is no significant accuracy reduction when using the template evolution approach.

The ETL_{CMT} system reduces the accuracy error by nearly 2.5% when compared to the single ETL system. The ETL_{CMT} accuracy, 96.72%, is the best one reported so far for the Mac-Morpho Corpus.

Table 5.3: System performances for the Tycho Brahe Corpus.

System	Accuracy (%)	# Templates
ETL _{CMT}	96.72	50
ETL	96.64	43
TBL	96.63	26
ETL _{TE}	96.60	43
BLS	93.00	–

5.6 TIGER Corpus

We have not found any work reporting the state-of-the-art performance for the TIGER Corpus. Therefore, for the TIGER Corpus, we report the performance of ETL, TBL and BLS systems.

In Table 5.4, we summarize the system performance results. The ETL system reduces the BLS accuracy error by 49%, from 6.69 to 3.43. ETL and TBL systems achieve similar accuracy. The ETL_{TE} system reduces the ETL training time by 61%, from 57 mins to 22 mins. Moreover, there is no accuracy reduction when using the template evolution approach. The ETL_{CMT} system reduces the accuracy error by nearly 3% when compared to the single ETL system.

Table 5.4: System performances for the TIGER Corpus.

System	Accuracy (%)	# Templates
ETL _{CMT}	96.68	50
ETL _{TE}	96.58	66
ETL	96.57	66
TBL	96.53	26
BLS	93.31	–

The TnT tagger [9], which is based on Hidden Markov Models, is a state-of-the-art system for German. It has been trained and tested on the NEGRA Corpus [68], which is the predecessor of the TIGER Corpus and uses the same tagset. For the NEGRA Corpus, Brants [9] reports an overall accuracy of 96.7%. Since both ETL and TBL systems performances for the TIGER Corpus are very close to the TnT performance for the NEGRA Corpus, we believe that both ETL and TBL systems achieve state-of-the-art performance for German POS tagging.

5.7 Brown Corpus

Eric Brill [13] presents a TBL system with state-of-the-art performance for the Brown Corpus. Therefore, for the Brown Corpus, we only report the performance of ETL, TBL and BLS systems.

In Table 5.5, we summarize the system performance results. The ETL system reduces the BLS accuracy error by 56%, from 7.57 to 3.31. ETL and TBL systems achieve similar accuracy. Therefore, ETL has state-of-the-art performance for the Brown Corpus. The ETL_{TE} system reduces the ETL training time by 68%, from 122 mins to 39 mins. Moreover, there is no significant accuracy reduction when using the template evolution approach. The ETL_{CMT} system reduces the accuracy error by 4% when compared to the single ETL system.

Table 5.5: System performances for the Brown Corpus.

System	Accuracy (%)	# Templates
ETL _{CMT}	96.83	50
ETL	96.69	63
TBL	96.67	26
ETL _{TE}	96.61	63
BLS	92.43	–

5.8 Summary

This chapter presents the application of ETL to language independent POS tagging. We apply ETL to four different corpora in three different languages. The selected corpora are: Mac-Morpho, a Portuguese language corpus; Tycho Brahe, a Portuguese language corpus; TIGER, a German language corpus; and Brown, an English language corpus.

Using the default parameter setting and a common set of features, the ETL system achieves state-of-the-art results for the four corpora. ETL and TBL achieve similar results. This finding indicates that the entropy guided template generation employed by ETL is very effective for the POS tagging task. It is interesting to note that the template set handcrafted by Brill has shown to be very robust for language independent POS tagging. The TBL system trained with this template set achieves very good results for the four corpora. This suggests that the useful feature combinations for POS tagging of the three languages are very similar. This hypothesis can be confirmed when looking at the template sets generated by ETL, which share many templates.

For instance, the template sets generated for Mac-Morpho and TIGER Corpus share 67% of the templates.

The rule application phase of ETL POS taggers is fast. For instance, our Python implementation of the ETL POS tagger created with the Mac-Morpho Corpus processes about 8,000 tokens per second.

The template evolution strategy provides a significant training time reduction in all cases. For the MAC-Morpho Corpus, the transformation learning is accelerated by a factor of almost four. Moreover, for all corpora, there is no significant accuracy reduction when using template evolution. This finding indicates that, for POS tagging, complex templates are not necessary for the earlier learned rules.

The ETL Committee strategy slightly improves the ETL accuracy for all corpora. However, the result is more significant only for the Mac-Morpho Corpus, where an accuracy error reduction of 6% is achieved. Training a committee of 100 classifiers using the Mac-Morpho Corpus takes about 13 hours on our 30 CPU-core cluster. While ETL Committee is an interesting method to improve accuracy without additional human effort, its classification time is very expensive, since many classifiers are used. It may only be useful when a computer cluster is available.

These chapter results indicate that ETL and ETL Committee are effective methods to produce competitive language independent POS taggers with little modeling effort.

6

Phrase Chunking

Phrase Chunking (PCK) consists in dividing a text into non-overlapping phrases [73]. It provides a key feature that helps on more elaborated NLP tasks such as NER and SRL. In the example that follows, we use brackets to indicate the eight phrase chunks in the sentence. In this example, there are four Noun Phrases (NP), two Verb Phrases (VP) and two Prepositional Phrases (PP).

```
[NP He ] [VP reckons ] [NP the current account  
deficit] [VP will narrow] [PP to ] [NP only  
# 1.8 billion ] [PP in ] [NP September ]
```

This chapter presents the application of the ETL approach to language independent PCK. We apply ETL to four different corpora in three different languages. In section 6.1, we describe the selected corpora. In section 6.2, we detail some modeling configurations used in our PCK system. In section 6.3, we show some configurations used in the machine learning algorithms. Section 6.4 presents the application of ETL for the SNR-CLIC Corpus. Section 6.5 presents the application of ETL for the Ramshaw & Marcus Corpus. Section 6.6 presents the application of ETL for the CoNLL-2000 Corpus. Section 6.7 presents the application of ETL for the SPSAL-2007 Corpus. Finally, section 6.8 presents some concluding remarks.

6.1 Corpora

We apply ETL to four different PCK corpora in three different languages. The selected corpora are: SNR-CLIC, a Portuguese NP chunking corpus [32]; Ramshaw & Marcus (R&M), an English base NP chunking corpus [59]; CoNLL-2000, an English phrase chunking corpus [73]; and SPSAL-2007, a Hindi phrase chunking corpus [7]. Table 6.1 shows some characteristics of these corpora.

The four corpora are annotated with *POS* and *PCK* tags. The *PCK tags* feature provides the phrase chunking annotation. For both corpora SNR-CLIC and Ramshaw & Marcus, the *PCK tags* feature uses the IOB1 tagging style, where: 0, means that the word is not a phrase chunk; I, means that the word

Table 6.1: Phrase Chunking Corpora.

Corpus	Language	Phrases	Training Data		Test Data	
			Sent.	Tokens	Sent.	Tokens
SNR-CLIC	Portuguese	NP	3,514	83,346	878	20,798
R&M	English	NP	8,936	211,727	2,012	47,377
CoNLL-2000	English	All	8,936	211,727	2,012	47,377
SPSAL-2007	Hindi	All	924	20,000	210	5,000

is part of a phrase chunk and B is used for the leftmost word of a phrase chunk beginning immediately after another phrase chunk. This tagging style is illustrated in the following noun phrase chunking example.

```
He/I reckons/O the/I current/I account/I deficit/I will/O
narrow/O to/O only/I #/I 1.8/I billion/I in/O September/I
```

For both corpora CoNLL-2000 and SPSAL-2007, the *PCK tags* feature uses the IOB2 tagging style, where: O, means that the word is not a phrase; B-X, means that the word is the first one of a phrase type X and I-X, means that the word is inside of a phrase type X. This tagging style is illustrated in the following phrase chunking example.

```
He/B-NP reckons/B-VP the/B-NP current/I-NP account/I-NP
deficit/I-NP will/B-VP narrow/I-VP to/B-PP only/B-NP #/I-NP
1.8/I-NP billion/I-NP in/B-PP September/B-NP
```

6.2 Phrase Chunking Modeling

When performing PCK, at both training and test time, we generate some *derived features*. This step is performed before the training and test time. It is described in the following subsection.

(a) Derived Features

The training and test corpora provide three input features: *word*, *POS* and *PCK tags*. Using the input features, we produce the following two derived features:

- **Capitalization:** this feature classifies the words according to their capitalization. It assumes one of the following categorical values: First Letter is Uppercase, All Letters are Uppercase, All Letters are Lowercase, Number, Punctuation, Number with “/” or “-” inside or Other.
- **Left verb:** for each token, this feature assumes the word feature value of the nearest predecessor verb;

6.3 Machine Learning Modeling

We use the following task specific configurations.

BLS: the baseline system assigns to each word the *PCK tag* that was most frequently associated with the part-of-speech of that word in the training set. The only exception was the initial classification of prepositions in the SNR-CLIC Corpus. In this case, the initial classification is done on an individual basis: each preposition has its frequency individually measured and the NP tag is assigned accordingly, in a lexicalized method.

TBL: in the TBL system we use a template set proposed by Ramshaw & Marcus [59]. This set contains 100 handcrafted templates which make use of the features *word*, *POS* and *PCK tags* in a context window of size 7. For the SNR-CLIC Corpus we extend this template set by adding the set of six special templates proposed by Santos & Oliveira [61]. The Santos & Oliveira’s six templates are designed to reduce classification errors of prepositions within the task of Portuguese noun phrase chunking. These templates use special handcrafted constraints that allow us to efficiently check the feature *word* in up to 20 left side adjacent tokens.

ETL: in the ETL learning, for the four corpora, we use the features *word*, *POS*, and *PCK tags*. The *left verb* feature is used only for the SNR-CLIC Corpus. The default ETL parameter setting shown in Section 4.2 is used. In order to make a fair comparison between ETL and TBL with handcrafted templates, we do not use the *capitalization* feature in the ETL system. Since this feature is not used in the Ramshaw & Marcus’s template set.

ETL_{TE}: we use the same ETL configuration, but here we perform template evolution.

ETL_{CMT}: we use the default ETL Committee parametrization shown in Section 4.3. Additionally we use the *capitalization* feature for the SNR-CLIC, Ramshaw & Marcus and CoNLL-2000 corpora.

We also report the state-of-the-art system performance for each corpus.

6.4 SNR-CLIC Corpus

Santos & Oliveira [61] present a TBL modeling that obtains state-of-the-art performance for Portuguese NP Chunking. The TBL system presented in this section uses the same modeling presented by Santos & Oliveira. For the SNR-CLIC Corpus, we report the performance of ETL, TBL and BLS systems.

In Table 6.2, we summarize the system performance results. The ETL system reduces the BLS $F_{\beta=1}$ error by 65%, from 31.94 to 11.15. Here, the ETL system outperforms the TBL system by 1.14 in terms of $F_{\beta=1}$. The ETL_{TE} system, which uses template evolution, has a training time 56% shorter than the one of ETL. However, there is a slightly performance loss in terms of $F_{\beta=1}$. The ETL_{CMT} system reduces the $F_{\beta=1}$ error by 6.6% when compared to the single ETL system.

Table 6.2: System performances for the SNR-CLIC Corpus.

System	Accuracy (%)	Precision (%)	Recall (%)	$F_{\beta=1}$	# Templates
ETL_{CMT}	98.09	89.66	89.51	89.58	50
ETL	97.97	88.77	88.93	88.85	46
ETL_{TE}	97.84	88.18	88.65	88.41	46
TBL	97.63	87.17	88.26	87.71	106
BLS	96.57	62.69	74.45	68.06	–

Observe that ETL generates only 46 template. However, this small template set is more effective than the handcrafted one, which has 106 templates. The number of templates generated by ETL is directly affected by the size of the training set. A small training set produces a small decision tree and, consequently, a small template set.

6.5 Ramshaw and Marcus Corpus

Kudo & Matsumoto [38] present an SVM-based system with state-of-the-art performance for the Ramshaw & Marcus Corpus. Therefore, for this Corpus, we also list the SVM system performance reported by Kudo & Matsumoto.

In Table 6.3, we summarize the system performance results. The ETL system reduces the BLS $F_{\beta=1}$ error by 63%, from 20.01 to 7.41. ETL slightly outperforms TBL in terms of $F_{\beta=1}$. The ETL_{TE} system reduces the ETL training time by 65%, from 21.31 mins to 7.43 mins. Moreover, the template evolution strategy slightly increases the ETL $F_{\beta=1}$. This finding suggests that,

for this corpus, complex templates are not necessary for the earlier learned rules.

The ETL_{CMT} system reduces the $F_{\beta=1}$ error by 9.4% when compared to the single ETL system. The ETL_{CMT} performance is competitive with the one of Kudo & Matsumoto’s SVM system.

Table 6.3: System performances for the Ramshaw & Marcus Corpus.

System	Accuracy (%)	Precision (%)	Recall (%)	$F_{\beta=1}$	# Templates
SVM	–	94.15	94.29	94.22	–
ETL_{CMT}	97.89	93.09	93.49	93.29	50
ETL_{TE}	97.61	92.56	93.04	92.80	106
ETL	97.52	92.49	92.70	92.59	106
TBL	97.42	91.68	92.26	91.97	100
BLS	94.48	78.20	81.87	79.99	–

6.6 CoNLL-2000 Corpus

Wu et al. [76] present an SVM-based system with state-of-the-art performance for the CoNLL-2000 Corpus. Therefore, for this Corpus, we also list the SVM system performance reported by Wu et al.

In Table 6.4, we summarize the system performance results. The ETL system reduces the BLS $F_{\beta=1}$ error by 66%, from 22.93 to 7.72. ETL, TBL and ETL_{TE} systems have similar performance results. However, the ETL_{TE} system reduces the ETL training time by 83%, from 160 mins to 30 mins. The ETL_{CMT} system significantly reduces the $F_{\beta=1}$ error by 13% when compared to the single ETL system. The ETL_{CMT} performance is competitive with the one of the SVM system.

Table 6.4: System performances for the CoNLL-2000 Corpus.

System	Accuracy (%)	Precision (%)	Recall (%)	$F_{\beta=1}$	# Templates
SVM	–	94.12	94.13	94.12	–
ETL_{CMT}	95.85	93.11	93.42	93.27	50
ETL	95.13	92.24	92.32	92.28	183
ETL_{TE}	95.03	91.89	92.28	92.09	183
TBL	95.12	92.05	92.28	92.16	100
BLS	77.29	72.58	82.14	77.07	–

In Table 6.5, we show the ETL_{CMT} system results, broken down by phrase chunk type, for the CoNLL-2000 Corpus.

Table 6.5: ETL_{CMT} results by chunk type for the CoNLL-2000 Corpus.

Chunk Type	Precision (%)	Recall (%)	F _{β=1}
ADJP	80.46	72.37	76.20
ADVP	80.81	78.75	79.77
CONJP	38.46	55.56	45.45
INTJ	0.00	0.00	0.00
LST	0.00	0.00	0.00
NP	93.41	93.86	93.63
PP	96.49	98.38	97.43
PRT	72.22	73.58	72.90
SBAR	90.75	86.17	88.40
VP	92.95	93.34	93.14
Overall	93.11	93.42	93.27

6.7 SPSAL-2007 Corpus

PVS & Gali [4] present a state-of-the-art system for the Hindi SPSAL-2007 Corpus. Their system uses a combination of Hidden Markov Models (HMM) and Conditional Random Fields (CRF). Therefore, for this Corpus, we also list the HMM+CRF system performance reported by PVS & Gali.

In Table 6.6, we summarize the system performance results. The results are reported in terms of chunking accuracy only, the same performance measure used in the SPSAL-2007 contest [7]. The ETL system reduces the BLS accuracy error by 28%, from 29.95 to 21.47. ETL and TBL systems achieve the same accuracy. The ETL_{TE} system reduces the ETL training time by 50%. However, there is a decrease in the system accuracy. The ETL_{CMT} system significantly reduces the accuracy error by 9% when compared to the single ETL system. The ETL_{CMT} performance is similar to the one of the HMM+CRF system.

Table 6.6: System performances for the SPSAL-2007 Corpus.

System	Accuracy (%)	# Templates
HMM + CRF	80.97	–
ETL _{CMT}	80.44	50
ETL	78.53	30
TBL	78.53	100
ETL _{TE}	77.21	30
BLS	70.05	–

6.8 Summary

This chapter presents the application of ETL to language independent phrase chunking. We apply ETL to four different corpora in three different languages. The selected corpora are: SNR-CLIC, a Portuguese noun phrase chunking corpus; Ramshaw & Marcus, an English base noun phrase chunking corpus; CoNLL-2000, an English phrase chunking corpus; and SPSAL-2007, a Hindi phrase chunking corpus.

Using the default parameter setting and a common set of features, the ETL system achieves state-of-the-art results for two corpora: SNR-CLIC and SPSAL-2007. For the other two, Ramshaw & Marcus and CoNLL-2000, the ETL system achieves state-of-the-art competitive results. ETL outperforms TBL with handcrafted templates in all the experiments. This finding indicates that the entropy guided template generation employed by ETL is very effective for the PCK task.

The rule application phase of ETL phrase chunkers is fast. For instance, our Python implementation of the ETL PCK created with the CoNLL-2000 Corpus processes about 12,000 tokens per second. The template evolution strategy provides a reasonable reduction of training time in all cases. For the CoNLL-2000 Corpus, the template evolution accelerates transformation learning by a factor of five. However, there is a slightly decrease in the ETL $F_{\beta=1}$.

ETL Committee significantly improves the ETL results for the four corpora. For the CoNLL-2000 Corpus, ETL Committee reduces the $F_{\beta=1}$ error by 13%, when compared to a single ETL model. Training a committee of 100 classifiers using the CoNLL-2000 Corpus takes about three and a half hours on our 30 CPU-core cluster.

These chapter results indicate that ETL and ETL Committee are effective methods to produce competitive language independent PCK systems with little modeling effort.

7

Named Entity Recognition

Named Entity Recognition (NER) is the problem of finding all proper nouns in a text and to classify them among several given categories of interest. Usually, there are three given categories: Person, Organization and Location. In the example that follows, we use brackets to indicate the four named entities in the sentence.

```
[PER Wolff ], currently a journalist in [LOC Argentina ],  
played with [PER Del Bosque ] in the final years of the  
seventies in [ORG Real Madrid ]
```

This chapter presents the application of the ETL approach to language independent NER. We apply ETL to three different corpora in three different languages. In section 7.1, we describe the selected corpora. In section 7.2, we detail some modeling configurations used in our NER system. In section 7.3, we show some configurations used in the machine learning algorithms. Section 7.4 presents the application of ETL for the HAREM Corpus. Section 7.5 presents the application of ETL for the SPA CoNLL-2002. Section 7.6 presents the application of ETL for the DUT CoNLL-2002. Finally, section 7.7 presents some concluding remarks.

7.1 Corpora

We apply ETL to three different NER corpora in three different languages. The selected corpora are: HAREM [64], a Portuguese NER corpus; SPA CoNLL-2002 [74], a Spanish NER corpus; and DUT CoNLL-2002 [74], a Dutch NER corpus. Table 7.1 shows some characteristics of the three selected corpora.

The HAREM Corpus is a golden set for NER in Portuguese [64]. This corpus is annotated with ten named entity categories: Person (PESSOA), Organization (ORGANIZACAO), Location (LOCAL), Value (VALOR), Date (TEMPO), Abstraction (ABSTRACCAO), Title (OBRA), Event (ACONTECIMENTO), Thing (COISA) and Other (VARIADO). Additionally, we au-

Table 7.1: Named Entity Recognition Corpora.

Corpus	Language	Training Data		Test Data	
		Sentenc.	Tokens	Sentenc.	Tokens
HAREM	Portuguese	4,749	98,475	3,393	66,627
SPA CoNLL-2002	Spanish	8,323	264,715	1,517	51,533
DUT CoNLL-2002	Dutch	15,806	202,931	5,195	68,994

tomatically generate the *POS* feature. We use the ETL POS tagger trained with the Mac-Morpho Corpus to create the *POS* feature. The HAREM corpus is already divided into two sets. Each set corresponds to a different Portuguese NER contest [64]. Here, we use the data from the first contest as the training set. The data from the second contest, which is called MiniHAREM, is our test set.

Both SPA CoNLL-2002 Corpus and DUT CoNLL-2002 Corpus were used in the CoNLL-2002 shared task [74]. They are annotated with four named entity categories: Person, Organization, Location and Miscellaneous. These corpora are also annotated with *POS* tags. The CONLL-2002 corpora are already divided into training and test sets. They also include development corpora which have characteristics similar to the test corpora. The development set is used to tune parameters of the learning systems.

In the three corpora, the *NE tags* feature provides the named entity annotation. The *NE tags* feature uses the IOB1 tagging style, where: O, means that the word is not a NE; I-X, means that the word is part of a NE type X and B-X is used for the leftmost word of a NE beginning immediately after another NE of the same type. The IOB1 tagging style is illustrated in the following example.

```
Wolff/I-PER ,/O currently/O a/O journalist/O in/O
Argentina/I-PLA ,/O played/O with/O Del/I-PER
Bosque/I-PER in/O the/O final/O years/O of/O
the/O seventies/O in/O Real/I-ORG Madrid/I-ORG
```

7.2 Named Entity Recognition Modeling

Our named entity recognition approach for the two CoNLL-2002 corpora follows the two stages strategy used in the training of ETL POS taggers (see Chapter 5). As in the POS tagging experiments, we use ETL for the learning of contextual rules only.

For the HAREM Corpus, we use ETL to classify only the five most frequent categories: Person, Organization, Location, Date and Value.

When performing NER, at both training and test time, we generate some *derived features*. This step is performed before the training and test time. It is described in the following subsection.

(a) Derived Features

The training and test corpora provide three input features: *word*, *POS* and *NE tags*. Using the input features, we produce the following four derived features:

- **Capitalization:** this feature classify the words according to their capitalization. It assumes one the following categorical values: First Letter is Uppercase, All Letters are Uppercase, All Letters are Lowercase, Number, Punctuation, Number with “/” or “-” inside or Other.
- **Dictionary membership:** this feature classify the words according to their presence in a dictionary. It assumes one the following categorical values: Upper, Lower, Both or None. For the corpora DUT CoNLL-2002 and SPA CoNLL-2002, the dictionary consists of the words appearing in their respective training sets. For the HAREM Corpus, the dictionary consists of the words appearing in the MAC-Morpho Corpus. We use the MAC-Morpho Corpus because its vocabulary is larger than the one of the HAREM Corpus.
- **Word Length:** this feature classify the words according to their lengths. It assumes one the following categorical values: 1, 2, 3, 4, 5, 6-8 or >8.
- **Noun Phrases:** this feature is generated using the ETL NP Chunker trained with the SNR-CLIC Corpus. We generate this feature only for the HAREM Corpus.

7.3 Machine Learning Modeling

The following ML model configurations provide our best results.

BLS: for the HAREM Corpus, we apply a BLS that makes use of gazetteers only. We use the gazetteers presented in [44], as well as some sections of the REPENTINO gazetteer [67]. From the REPENTINO gazetteer we use only the categories Beings, Location and Organization. We use only some subcategories of the REPENTINO gazetteer. From category Beings we use subcategory Human. From Location we use Terrestrial, Town, Region and Adm. Division. From Organization we use all subcategories. For some subcategories an extra processing is

also required. From Human, we extract only first names. From the Organization category, we use full company names and extract the top 100 most frequent words. We also use a month name gazetteer and a list of lower case words that can start a NE.

For both SPA and DUT CoNLL 2002 corpora, the baseline system assigns to each word the *NE tag* that was most frequently associated with that word in the training set. If capitalized, an unknown word is tagged as a person, otherwise it is tagged as non entity.

TBL: for the HAREM Corpus, the reported results for the TBL approach refer to TBL trained with the 32 handcrafted template set proposed by Milidiú et al. [44]. For the two CoNLL 2002 corpora, we use the Brill’s template set [13].

ETL: in the ETL learning, for the three corpora, we use the features *word*, *POS*, *NE tags*, *capitalization information*, *dictionary membership* and *word length*. For the HAREM Corpus, we additionally use the *noun phrase* feature. The default ETL parameter setting shown in Section 4.2 is used.

ETL_{TE}: we use the same ETL configuration, but here we perform template evolution.

ETL_{CMT}: we use the default ETL Committee parametrization shown in Section 4.3.

We also report the state-of-the-art system performance for each corpus.

In order to assess the system performances over the two CoNLL-2002 corpora, we use the evaluation tool provided in the CoNLL-2002 website [74]. For the HAREM Corpus, we use the evaluation tools described in [64].

7.4 HAREM Corpus

As Santos & Cardoso [64], we report the category classification results in two scenarios: total and selective. In the *total scenario*, all the categories are taken into account when scoring the systems. In the *selective scenario*, only the five chosen categories (Person, Organization, Location, Date and Value) are taken into account.

Here, our test corpus corresponds to the data used in the second Portuguese NER contest, the MiniHAREM [64]. For this test corpus, the CORTEX system [3] shows the best result reported so far. According to Aranha [3], the CORTEX system relies in the use of handcrafted rules that jointly work with a

rich knowledge base for the NER task. Here, we also list the CORTEX system performance reported by Aranha.

In tables 7.2 and 7.3, we summarize the system performance results for the total and selective scenarios, respectively. For the total scenario, the ETL system reduces the BLS $F_{\beta=1}$ error by 39%, from 64.26 to 39.18. In both scenarios, ETL and CORTEX have similar results. In both tables, we can see that ETL significantly outperforms the TBL system. Using template evolution, the training time is reduced by nearly 35% and there is no performance loss. Actually, the ETL_{TE} system has a $F_{\beta=1}$ slightly better than the one of ETL. In both scenarios, ETL_{CMT} achieves the best $F_{\beta=1}$. In the total scenario, the ETL_{CMT} system reduces the $F_{\beta=1}$ error by 7% when compared to the single ETL system. As far as we know, ETL_{CMT} results are the best reported so far for named entity classification of the HAREM Corpus.

Table 7.2: System performances in the Total Scenario for the HAREM Corpus.

System	Precision (%)	Recall (%)	$F_{\beta=1}$	# Templates
ETL_{CMT}	77.52	53.86	63.56	50
CORTEX	77.85	50.92	61.57	–
ETL_{TE}	74.92	51.90	61.32	87
ETL	71.08	53.15	60.82	87
TBL	57.78	45.20	50.72	32
BLS	47.87	28.52	35.74	–

Table 7.3: System performances in the Selective Scenario for the HAREM Corpus.

System	Precision (%)	Recall (%)	$F_{\beta=1}$	# Templates
ETL_{CMT}	77.27	65.20	70.72	50
CORTEX	77.86	60.97	68.39	–
ETL_{TE}	74.75	62.82	68.27	87
ETL	70.90	64.34	67.46	87
TBL	57.76	54.69	56.19	32
BLS	47.85	34.52	40.11	–

The TBL results for the HAREM Corpus are very poor due to its template set, which is the same used with the LearnNEC06 Corpus proposed in [44]. This is the only TBL template set that we have found for the Portuguese NER task. This template set contains some pre-instantiated tests, therefore it seems to be very restricted to the kind of named entity structures that appear in the LearnNEC06 Corpus.

In Table 7.4, we show the ETL_{CMT} system results, broken down by named entity type, for the HAREM Corpus. The best result is obtained for the *Date* category, where a precision of 88.29% is achieved. This is mainly because most of the NEs in this category follow very simple and fixed patterns such as “the [a day] of [a month]”. The NE *Organization* is the most difficult one to extract. We believe that this difficulty comes mainly from two facts: (1) the length of organization names is more dynamic. For instance, in the HAREM Corpus some organization names are composed by more than seven words; and (2) the recognition of organization name abbreviations is a hard task.

Table 7.4: ETL_{CMT} results by entity type for the HAREM Corpus.

Entity	Precision (%)	Recall (%)	$F_{\beta=1}$
Date	88.29	82.21	85.14
Location	76.18	68.16	71.95
Organization	65.34	50.29	56.84
Person	81.49	61.14	69.87
Vale	77.72	70.13	73.73
Overall	77.27	65.20	70.72

Like any other ML based strategy, the versatility of ETL based systems is one advantage over other non-ML based systems, such as CORTEX. All the used resources, but the training set and gazetteers, are language independent. As we show in the next two sections, we can quickly create an ETL based NER system for any language that has an available training set.

7.5 SPA CoNLL-2002 Corpus

Carreras et al. [16] present an AdaBoost system with state-of-the-art performance for the SPA CoNLL-2002 Corpus. Their AdaBoost system uses decision trees as a base learner. Therefore, for the SPA CoNLL-2002 Corpus, we also list the AdaBoost system performance reported by Carreras et al.

In Table 7.5, we summarize the system performance results. The ETL system reduces the BLS $F_{\beta=1}$ error by 47%, from 44.49 to 23.72. For the SPA CoNLL-2002 Corpus, ETL significantly outperforms TBL. The ETL_{TE} system reduces the ETL training time by 57%, from 26.3 mins to 11.3 mins. However, there is a performance loss in terms of $F_{\beta=1}$. The ETL_{CMT} system reduces the $F_{\beta=1}$ error by 5% when compared to the single ETL system.

Through a quick analysis of the results, we found out that the majority of errors are related to *names* that do not occur in the training corpus. This

Table 7.5: System performances for the SPA CoNLL-2002 Corpus.

System	Precision (%)	Recall (%)	$F_{\beta=1}$	# Templates
AdaBoost	79.27	79.29	79.28	–
ETL _{CMT}	76.99	77.94	77.46	50
ETL	75.50	77.07	76.28	98
ETL _{TE}	74.28	75.64	74.95	98
TBL	72.27	74.99	73.61	26
BLS	49.59	63.02	55.51	–

result suggests that the use of gazetteers has the potential to improve the ETL results.

Although the ETL system has a $F_{\beta=1}$ smaller than the one of the AdaBoost system, the ETL modeling phase seems to be simpler than the one of the Carreras et al.’s system. They divide the NER task into two intermediate tasks: NE identification and NE classification. In the first stage, the AdaBoost system identifies NE candidates. In the second stage, the AdaBoost system classify the identified candidates. Our ETL approach for NER is more straightforward, since we do not divide the task. Nevertheless, for the SPA CoNLL-2002, the ETL Committee system is in top three when compared with the 12 CoNLL-2002 contestant systems.

In Table 7.6, we show the ETL_{CMT} system results, broken down by named entity type, for the SPA CoNLL-2002 Corpus.

Table 7.6: ETL_{CMT} results by entity type for the SPA CoNLL-2002 Corpus.

Entity	Precision (%)	Recall (%)	$F_{\beta=1}$
Location	80.75	75.83	78.21
Miscellaneous	54.06	50.88	52.42
Organization	76.01	80.57	78.22
Person	83.35	88.57	85.88
Overall	76.99	77.94	77.46

7.6 DUT CoNLL-2002 Corpus

The Carreras et al. [16] AdaBoost based system is also a state-of-the-art system for the DUT CoNLL-2002 Corpus. Therefore, for the DUT CoNLL-2002 Corpus, we also list the AdaBoost system performance reported by Carreras et al.

In Table 7.7, we summarize the system performance results. The ETL system reduces the BLS $F_{\beta=1}$ error by 47%, from 48.58 to 25.82. For the DUT CoNLL-2002 Corpus, ETL significantly outperforms TBL. The ETL_{TE} system reduces the ETL training time by 55%, from 15.5 mins to 7 mins. However, there is a performance loss in terms of $F_{\beta=1}$. The ETL_{CMT} system reduces the $F_{\beta=1}$ error by 5% when compared to the single ETL system.

Table 7.7: System performances for the DUT CoNLL-2002 Corpus.

System	Precision (%)	Recall (%)	$F_{\beta=1}$	# Templates
AdaBoost	77.83	76.29	77.05	–
ETL_{CMT}	76.52	74.38	75.44	50
ETL	74.97	73.39	74.18	79
ETL_{TE}	73.96	72.33	73.14	79
TBL	69.42	72.05	70.71	26
BLS	47.68	55.80	51.42	–

As in the SPA CoNLL-2002 Corpus, the majority of errors are related to *names* that do not occur in the training corpus. Therefore, we believe that the use of gazetteers has also the potential to improve the ETL results for the DUT CoNLL-2002 Corpus.

Similarly to the SPA CoNLL-2002 Corpus, the Carreras et al.’s system divide the NER task into two intermediate tasks: NE identification and NE classification. In the first stage, the AdaBoost system identifies NE candidates. In the second stage, the AdaBoost system classify the identified candidates. Our ETL approach for NER is more straightforward, since we do not divide the task. Nevertheless, for the DUT CoNLL-2002, the ETL Committee system is in top two when compared with the 12 CoNLL-2002 contestant systems.

In Table 7.8, we show the ETL_{CMT} system results, broken down by named entity type, for the DUT CoNLL-2002 Corpus.

Table 7.8: ETL_{CMT} results by entity type for the DUT CoNLL-2002 Corpus.

Entity	Precision (%)	Recall (%)	$F_{\beta=1}$
Location	79.77	80.80	80.28
Miscellaneous	72.51	69.33	70.89
Organization	78.80	62.71	69.84
Person	76.87	84.60	80.55
Overall	76.52	74.38	75.44

7.7 Summary

This chapter presents the application of ETL to language independent named entity recognition. We apply ETL to three different corpora in three different languages. The selected corpora are: HAREM, a Portuguese NER corpus; SPA CoNLL-2002, a Spanish NER corpus; and DUT CoNLL-2002, a Dutch NER corpus.

Using the default parameter setting and a common set of features, the ETL system achieves state-of-the-art competitive results for the three corpora. ETL outperforms TBL with handcrafted templates in all the experiments. This finding indicates that the entropy guided template generation employed by ETL is very effective for the NER task.

The rule application phase of ETL NER systems is fast. For instance, our Python implementation of the ETL NER created with the HAREM Corpus processes about 9,500 tokens per second. The template evolution strategy provides a reasonable reduction of training time in the three corpora. However, for both SPA CoNLL-2002 Corpus and DUT CoNLL-2002 Corpus the template evolution reduces the ETL $\mathbf{F}_{\beta=1}$.

ETL Committee significantly improves the ETL results for the three corpora. For the HAREM Corpus, ETL Committee achieves the best result reported so far. Training a committee of 100 classifiers using the SPA CoNLL-2002 Corpus takes about two hours on our 30 CPU-core cluster.

These chapter results indicate that ETL and ETL Committee are effective methods to produce competitive language independent NER systems with little modeling effort.

8

Semantic Role Labeling

Semantic Role Labeling (SRL) is the process of detecting basic event structures such as *who* did *what* to *whom*, *when* and *where* [70]. The SRL task is performed at the sentence-level. More specifically, for each predicate of a clause, whose head is typically a verb, all the constituents in the sentence which fill a semantic role of the verb have to be recognized. Typical semantic roles, also called arguments, include Agent, Patient, Instrument, and also adjuncts such as Locative, Temporal, Manner, and Cause [17]. A verb and its set of arguments form a *proposition* in the sentence. SRL provides a key knowledge that helps to build more elaborated document management and information extraction applications. The following sentence illustrates this labeling task.

[A0 He] [AM-MOD would] [AM-NEG n't] [V accept] [A1
anything of value] from [A2 those he was writing about] .

Here, the semantic roles of the predicate *accept* are marked with the labels: V=*verb*; A0=*acceptor*; A1=*thing accepted*; A2=*accepted-from*; AM-MOD=*modal*; AM-NEG=*negation*.

This chapter presents the application of the ETL approach to SRL. We evaluate the performance of ETL over two English language corpora. In section 8.1, we describe the selected corpora. In section 8.2, we detail some modeling configurations used in our SRL system. In section 8.3, we show some configurations used in the machine learning algorithms. Section 8.4 presents the application of ETL for the CoNLL-2004 Corpus. Section 8.5 presents the application of ETL for the CoNLL-2005 Corpus. Finally, section 8.6 presents some concluding remarks.

8.1 Corpora

We evaluate the performance of ETL over two English language corpora: CoNLL-2004 [17] and CoNLL-2005 [18]. These two corpora were used in the CoNLL shared task of the years 2004 and 2005, respectively. Table 8.1 shows some characteristics of these corpora.

Table 8.1: Semantic Role Labeling Corpora.

Corpus	Language	Training Data		Test Data	
		Sentenc.	Tokens	Sentenc.	Tokens
CoNLL-2004	English	8,936	211,727	1,671	40,039
CoNLL-2005	English	39,832	950,028	2,416	56,684

The CoNLL-2005 Corpus consists of the Proposition Bank (PropBank) [53], an approximately one-million-word corpus annotated with predicate-argument structures. The PropBank annotates the Wall Street Journal part of the Penn TreeBank [41] with verb argument structure. The CoNLL-2005 Corpus is already divided into training and test sets. The Penn TreeBank sections 02-21 are used for training. Additionally, section 23 is used for test. Section 24 is used as a development set, to tune parameters of the learning systems. The CoNLL-2004 Corpus is a subset of the CoNLL-2005 Corpus. It uses Penn TreeBank sections 15-18 for training and section 21 for test. In the CoNLL-2004 Corpus, section 20 is used as a development set.

The SRL corpora are annotated with four basic input features: *POS tags*, *phrase chunks*, *clauses* and *named entities*. *Clauses* are word sequences which contain a subject and a predicate. The *clause* feature indicates when a token begins or ends a clause. These four input features were automatically generated using state-of-the-art systems [18]. The SRL corpora also include two other features: the *target verbs* feature, which indicates the verbs whose arguments must be labeled; and the *srl tags* feature, which provides the semantic labeling.

The *srl tags* used in the PropBank annotation numbers the arguments of each predicate from A0 to A5. Usually, A0 stands for agent, A1 for theme or direct object, and A2 for indirect object, benefactive or instrument, but semantics tend to be verb specific [70]. Adjunctive arguments are referred to as AM-T, where T is the type of the adjunct. For instance, AM-MNR indicates manner, AM-TMP indicates a temporal, and AM-LOC indicates a locative. Arguments can be discontinuous, in which case the continuation fragments are prefixed by C-, for instance, “[A1 *The economy*], [A0 *he*] [V *noted*], [C-A1 *moves the market*], *not vice versa*”. In the PropBank annotation, argument references share the same label with the actual argument prefixed with R-. References are typically pronominal.

In Figure 8.1, we show an annotated sentence from the CoNLL 2004 Corpus. The input columns consists of words (1st), POS tags (2nd), base chunks (3rd), clauses (4th) and named entities (5th). The 6th column marks target verbs, and their propositions are found in 7th and 8th columns.

According to the PropBank annotation, for *stew* (7th), the A0 annotates the *worrier*, and the A1 the *cause*. For *put* (8th), the A0 annotates the *putter*, A1 is the *thing put*, and A2 the *where put*.

Bob	NNP	B-NP	(S*	B-PER	-	(A0*	*
Stone	NNP	I-NP	*	I-PER	-	*A0)	*
stewed	VBD	B-VP	*	0	stew	(V*V)	*
over	IN	B-PP	*	0	-	*	*
a	DT	B-NP	*	0	-	(A1*	(A0*
letter	NN	I-NP	*	0	-	*	*
from	IN	B-PP	*	0	-	*	*
his	PRP\$	B-NP	*	0	-	*	*
manager	NN	I-NP	*	0	-	*	*A0)
putting	VBG	B-VP	*	0	put	*	(V*V)
him	PRP	B-NP	*	0	-	*	(A1*A1)
on	IN	B-PP	*	0	-	*	*
probation	NN	B-NP	*	0	-	*	(A2*A2)
for	IN	B-PP	*	0	-	*	(AM-CAU*
insubordination	NN	B-NP	*	0	-	*A1)	*AM-CAU)
.	.	0	*S)	0	-	*	*

Figure 8.1: Example of an annotated sentence from the CoNLL 2004 Corpus.

The CoNLL-2005 Corpus also includes *full parsing* features, the complete syntactic trees given by two alternative parsers. This kind of feature eases the semantic role labeling task [1]. However, since our purpose is to examine the ETL performance for a complex task, we do not use the full parsing features in our SRL experiments. Therefore, the same system configuration is used for both corpora.

8.2 Semantic Role Labeling Modeling

We approach the SRL task as a token classification problem. Which means that, for each token, the system must decide when the token starts or ends a specific kind of argument.

When performing SRL, at both training and test time, we have three additional steps: generation of derived features, preprocessing and postprocessing. The preprocessing and generation of derived features are performed before the training and test time. The postprocessing is executed after the training and test time. These three steps are described in the following subsections.

(a) Derived Features

The training and test corpora provide six input features: *word*, *POS*, *named entities*, *phrase chunks*, *clauses*, *target verbs* and *SRL tags*. Using the input features, we produce the following thirteen derived features:

- **Token Position:** indicates if the token comes before or after the target verb.
- **Temporal:** indicates if the word is or not a temporal keyword. The temporal keywords list contains the words that frequently appear as temporal adjuncts in the training set, such as the words *December*, *Monday* and *since*.
- **Path:** the sequence of chunk tags between the chunk and the target verb. Consecutive NP chunks are treated as one.
- **Pathlex:** the same as the *path* feature with the exception that here we use the preposition itself instead of the PP chunk tag.
- **Distance:** the number of chunks between the chunk and the target verb. Consecutive NP chunks are also treated as one.
- **VP Distance:** distance, in number of VP chunks, between the token and the target verb.
- **Clause Path:** the clause bracket chain between the token and the target verb.
- **Clause Position:** indicates if the token is inside or outside of the clause which contains the target verb.
- **Number of Predicates:** number of target verbs in the sentence.
- **Voice:** indicates the target verb voice: active or passive.
- **Target Verb POS:** POS tag of the target verb.
- **Predicate POS Context:** the POS tags of the words that immediately precede and follow the predicate. The POS tag of a preposition is replaced with the preposition itself.
- **Predicate Argument Patterns:** for each predicate, we identify the most frequent left and right patterns of the core arguments (A0 through A5) in the training set. For instance, *separate* has A0 and A1_A2 as its most frequent left and right argument patterns, respectively. For the *separate* verb, this feature assumes the value A0 for the tokens on the left side of the verb, and the value A1_A2 for the tokens on the right side.

All these features were previously used in other SRL systems [34, 42]. For those systems, these features prove to be very important for performance improvement.

(b) Preprocessing

We use a different token representation for the SRL task. Here, our system classifies phrase chunks instead of words. Which means that here a token represents a complete phrase chunk. In the preprocessing step, the original word-based tokens are collapsed in order to generate the new representation. In the collapsing process, only the feature values of the phrase chunk headwords are retained. The chunk headword is defined as its rightmost word. This preprocessing speeds up the training step, since the number of tokens to be annotated are reduced. As a consequence, larger sentence segments are covered with smaller context window sizes. In Figure 8.2 we show the collapsed version of the sentence shown in Figure 8.1. In this example, the chunks “*Bob Stone*”, “*a letter*” and “*his manager*” are collapsed.

<i>Stone</i>	NNP	I-NP	(S*	I-PER	-	(AO*AO)	*
<i>stewed</i>	VBD	B-VP	*	0	<i>stew</i>	(V*V)	*
<i>over</i>	IN	B-PP	*	0	-	*	*
<i>letter</i>	NN	I-NP	*	0	-	(A1*	(AO*
<i>from</i>	IN	B-PP	*	0	-	*	*
<i>manager</i>	NN	I-NP	*	0	-	*	*AO)
<i>putting</i>	VBG	B-VP	*	0	<i>put</i>	*	(V*V)
<i>him</i>	PRP	B-NP	*	0	-	*	(A1*A1)
<i>on</i>	IN	B-PP	*	0	-	*	*
<i>probation</i>	NN	B-NP	*	0	-	*	(A2*A2)
<i>for</i>	IN	B-PP	*	0	-	*	(AM-CAU*
<i>insubordination</i>	NN	B-NP	*	0	-	*A1)	*AM-CAU)
.	.	0	*S)	0	-	*	*

Figure 8.2: Example of a sentence in the collapsed tokens version.

We treat propositions independently. Therefore, for each target verb we generate a separate sequence of tokens to be annotated. In general, all the arguments of a proposition are inside the target verb clause. Hence, we do not include tokens that are outside of the target verb clause. The only exception is when we have a nested clause that begins with a target verb. Here, we must also include the external clause.

In Figure 8.3, we show the token sequences for the two target verbs shown in Figure 8.2. Note that, for this particular case, the token sequences for each verb are the same. Only the two last columns, which defines the target verb and the proposition annotation, are verb specific.

(c) Postprocessing

During the postprocess step, some constraints are added to the extracted arguments. These constraints remove simple mistakes by removing or editing

<i>Stone</i>	NNP	I-NP	(S*	I-PER	-	(A0*AO)
<i>stewed</i>	VBD	B-VP	*	0	<i>stew</i>	(V*V)
<i>over</i>	IN	B-PP	*	0	-	*
<i>letter</i>	NN	I-NP	*	0	-	(A1*
<i>from</i>	IN	B-PP	*	0	-	*
<i>manager</i>	NN	I-NP	*	0	-	*
<i>putting</i>	VBG	B-VP	*	0	-	*
<i>him</i>	PRP	B-NP	*	0	-	*
<i>on</i>	IN	B-PP	*	0	-	*
<i>probation</i>	NN	B-NP	*	0	-	*
<i>for</i>	IN	B-PP	*	0	-	*
<i>insubordination</i>	NN	B-NP	*	0	-	*A1)
.	.	0	*S)	0	-	*
<i>Stone</i>	NNP	I-NP	(S*	I-PER	-	*
<i>stewed</i>	VBD	B-VP	*	0	-	*
<i>over</i>	IN	B-PP	*	0	-	*
<i>letter</i>	NN	I-NP	*	0	-	(AO*
<i>from</i>	IN	B-PP	*	0	-	*
<i>manager</i>	NN	I-NP	*	0	-	*A0)
<i>putting</i>	VBG	B-VP	*	0	<i>put</i>	(V*V)
<i>him</i>	PRP	B-NP	*	0	-	(A1*A1)
<i>on</i>	IN	B-PP	*	0	-	*
<i>probation</i>	NN	B-NP	*	0	-	(A2*A2)
<i>for</i>	IN	B-PP	*	0	-	(AM-CAU*
<i>insubordination</i>	NN	B-NP	*	0	-	*AM-CAU)
.	.	0	*S)	0	-	*

Figure 8.3: Token sequences for the target verbs *stew* and *put*.

identified arguments. The following constraints and corresponding editions are implemented.

- There can be only one argument A[0-5] per sentence. If more than one is found, we keep only the target verb closest argument, all the others being removed.
- An argument can not have its boundaries outside the clause it starts. Otherwise, the argument would have its boundaries cropped to fit the clause boundaries.

8.3 Machine Learning Modeling

We use the following task specific configurations.

BLS: we use the same baseline system proposed for the CoNLL-2004 Shared Task [17]. It is based on six heuristic rules that make use of POS and phrase chunks:

1. Tag *not* and *n't* in target verb chunk as AM-NEG.
2. Tag *modal verbs* in target verb chunk as AM-MOD.

3. Tag first NP before target verb as A0.
4. Tag first NP after target verb as A1.
5. Tag *that*, *which* and *who* before target verb as R-A0.
6. Switch A0 and A1, and R-A0 and R-A1 if the target verb is part of a passive VP chunk.

TBL: we do not use handcrafted TBL templates for the SRL task. Therefore, we only present the TBL results for the CoNLL-2004 corpora reported by Higgins [35]. Higgins used a set of 133 templates which reference basic and derived features.

ETL: in the ETL learning, we use all the input and derived features. The default ETL parameter setting shown in Section 4.2 is used.

ETL_{TE}: we use the same ETL configuration, but here we perform template evolution.

ETL_{CMT}: we use the default ETL Committee parametrization shown in Section 4.3.

We also report the state-of-the-art system performance for each corpus.

In order to assess the system performances on the CoNLL-2004 Corpus, we use the evaluation tool provided in the CoNLL-2004 website [17]. For the CoNLL-2005 Corpus, we use the evaluation tool provided in the CoNLL-2005 website [18].

8.4 CoNLL-2004 Corpus

Hacioglu et al. [34] present a SVM-based system that does not use full parsing. This system achieves state-of-the-art performance for the CoNLL-2004 Corpus. Therefore, for this Corpus, we also list the SVM system performance reported by Hacioglu et al.

In Table 8.2, we summarize the system performance results. The ETL system reduces the BLS $F_{\beta=1}$ error by 40%, from 60.55 to 36.63. Our ETL system reduces the $F_{\beta=1}$ error by 7% when compared to the Higgins’s TBL system. The ETL_{TE} system reduces the ETL training time by 78%, from 10 hours to 2.2 hours. However, there is a performance loss in terms of $F_{\beta=1}$. The lengthy training time is due to the large number of generated templates. For this corpus, ETL generates 632 templates.

The postprocessing step has very little impact in the final performance. For the ETL system, the postprocessing improves the $F_{\beta=1}$ by only 0.3.

The ETL_{CMT} system reduces the $F_{\beta=1}$ error by 11% when compared to the single ETL system. ETL_{CMT} performance is very competitive with the one of the SVM system. Moreover, the precision error of the ETL_{CMT} system is 15% smaller than the one of the SVM system. As far as we know, the precision of the ETL_{CMT} , 76.44, is the best one reported so far for a system that does not use full or joint parsing. In most situations, precision is more important than recall. Nevertheless, for the CoNLL-2004, the ETL Committee system is in top two when compared with the 10 CoNLL-2004 contestant systems.

Table 8.2: System performances for the CoNLL-2004 Corpus.

System	Precision (%)	Recall (%)	$F_{\beta=1}$	# Templates
SVM	72.43	66.77	69.49	–
ETL_{CMT}	76.44	60.25	67.39	50
ETL	70.60	57.48	63.37	632
ETL_{TE}	69.41	56.85	62.50	632
TBL	64.17	57.52	60.66	130
BLS	55.57	30.58	39.45	–

In Table 8.3 we show the ETL_{CMT} system results, broken down by argument type, for the CoNLL-2004 Corpus. Among the core arguments, A1–A5, the best results are for A0 and A1, which are the most frequent ones. Among the adjuncts, the best results are for AM–MOD, modal, and AM–NEG, negation, which follow very simple fixed patterns.

8.5 CoNLL-2005 Corpus

Surdeanu et al. [70] present an AdaBoost-based system that does not use full parsing. This system, as far as we know, achieves the best reported performance for the CoNLL-2005 Corpus when full parsing is not considered. Therefore, for this Corpus, we also list the AdaBoost system performance reported by Surdeanu et al.

Due to memory restrictions, we train the ETL system using templates that combine at most five features. However, for the ETL_{TE} system we can use all the templates, since this training strategy uses less memory. In Table 8.2, we summarize the system performance results. The ETL system reduces the BLS $F_{\beta=1}$ error by 53%, from 63.86 to 29.92. Even using a larger template set, the ETL_{TE} system reduces the ETL training time by 63%. Nevertheless, there is a performance loss in terms of $F_{\beta=1}$. This reduction in the $F_{\beta=1}$ indicates that the SRL task requires complex templates throughout the training process.

Table 8.3: ETL_{CMT} results by argument type for the CoNLL-2004 Corpus.

	Precision	Recall	F _{β=1}
Overall	76.44%	60.25%	67.39
A0	86.19%	68.98%	76.63
A1	75.69%	66.84%	70.99
A2	59.92%	43.98%	50.73
A3	52.63%	33.33%	40.82
A4	70.73%	58.00%	63.74
A5	0.00%	0.00%	0.00
AA	0.00%	0.00%	0.00
AM-ADV	53.09%	28.01%	36.67
AM-CAU	40.00%	16.33%	23.19
AM-DIR	32.00%	16.00%	21.33
AM-DIS	71.35%	57.28%	63.54
AM-EXT	57.14%	57.14%	57.14
AM-LOC	46.49%	23.25%	30.99
AM-MNR	56.36%	24.31%	33.97
AM-MOD	99.68%	92.58%	96.00
AM-NEG	94.57%	96.06%	95.31
AM-PNC	44.44%	9.41%	15.53
AM-PRD	0.00%	0.00%	0.00
AM-TMP	73.08%	48.33%	58.18
R-A0	85.62%	78.62%	81.97
R-A1	78.72%	52.86%	63.25
R-A2	44.44%	44.44%	44.44
R-A3	0.00%	0.00%	0.00
R-AA	0.00%	0.00%	0.00
R-AM-LOC	100.00%	25.00%	40.00
R-AM-MNR	0.00%	0.00%	0.00
R-AM-PNC	0.00%	0.00%	0.00
R-AM-TMP	66.67%	14.29%	23.53
V	98.07%	98.07%	98.07

The reason is that SRL is a hard task which makes use of a large number of features.

The ETL_{CMT} system reduces the F_{β=1} error by 7.2% when compared to the single ETL system. The ETL_{CMT} performance is competitive with the one of the AdaBoost system. Again, the ETL_{CMT} presents state-of-the-art precision performance. It is important to note that we do not use full parsing in our experiments. State-of-the-art systems using full parsing achieve F_{β=1} of about 80% in the CoNLL-2005 test set.

In Table 8.5 we show the ETL_{CMT} system results, broken down by argument type, for the CoNLL-2005 Corpus.

Table 8.4: System performances for the CoNLL-2005 Corpus.

System	Precision (%)	Recall (%)	$F_{\beta=1}$	# Templates
AdaBoost	78.76	72.44	75.47	–
ETL _{CMT}	80.54	65.47	72.23	50
ETL	76.79	64.45	70.08	434
ETL _{TE}	74.75	63.27	68.53	632
BLS	51.33	27.88	36.14	–

Table 8.5: ETL_{CMT} results by argument type for the CoNLL-2005 Corpus.

datalabel	Precision	Recall	$F_{\beta=1}$
Overall	80.54%	65.47%	72.23
A0	88.47%	70.84%	78.68
A1	78.43%	69.88%	73.91
A2	67.12%	57.57%	61.98
A3	69.23%	41.62%	51.99
A4	70.83%	66.67%	68.69
A5	50.00%	100.00%	66.67
AA	0.00%	0.00%	0.00
AM-ADV	71.90%	34.39%	46.52
AM-CAU	70.97%	30.14%	42.31
AM-DIR	53.97%	40.00%	45.95
AM-DIS	82.04%	62.81%	71.15
AM-EXT	72.73%	50.00%	59.26
AM-LOC	63.93%	38.57%	48.11
AM-MNR	64.47%	42.73%	51.40
AM-MOD	98.86%	94.19%	96.47
AM-NEG	99.10%	96.09%	97.57
AM-PNC	60.53%	20.00%	30.07
AM-PRD	0.00%	0.00%	0.00
AM-REC	0.00%	0.00%	0.00
AM-TMP	80.10%	58.51%	67.62
R-A0	90.26%	78.57%	84.01
R-A1	78.99%	69.87%	74.15
R-A2	58.33%	43.75%	50.00
R-A3	0.00%	0.00%	0.00
R-A4	0.00%	0.00%	0.00
R-AM-ADV	0.00%	0.00%	0.00
R-AM-CAU	100.00%	25.00%	40.00
R-AM-EXT	0.00%	0.00%	0.00
R-AM-LOC	85.71%	28.57%	42.86
R-AM-MNR	50.00%	33.33%	40.00
R-AM-TMP	79.49%	59.62%	68.13
V	98.90%	98.90%	98.90

8.6 Summary

This chapter presents the application of the ETL approach to SRL. We evaluate the performance of ETL over two English language corpora: CoNLL-2004 and CoNLL-2005.

Using the default parameter setting and a common set of features, the ETL system achieves regular results for the two corpora. However, for the CoNLL-2004 Corpus, our ETL system outperforms the TBL system proposed by Higgins [35]. This finding indicates that, for the SRL task, the entropy guided template generation employed by ETL is more effective than the Higgins’s handcrafted templates.

The classification phase of the ETL SRL systems is very fast. For instance, our Python implementation of the ETL SRL system created with the CoNLL-2005 Corpus performs the whole classification of the test set in only 30 seconds. The template evolution strategy provides a significant reduction of training time in the two cases. However, there is a slightly decrease in the ETL $F_{\beta=1}$.

ETL Committee significantly improves the ETL results for the two corpora. When compared to the single ETL systems, ETL Committee reduces the $F_{\beta=1}$ error by 11% and 7% for the CoNLL-2004 and CoNLL-2005 Corpus, respectively. We believe that the error reduction is larger for the CoNLL-2004 Corpus because this corpus is smaller. Training a committee of 100 classifiers using the CoNLL-2004 Corpus takes about ten hours on our 30 CPU-core cluster.

These chapter results indicate that ETL Committee is an effective method to produce very competitive and precise SRL systems with little modeling effort. Moreover, using the ETL approach, we can produce regular SRL systems that work very fast.

9

Conclusions

Entropy Guided Transformation Learning is a new machine learning algorithm for classification tasks. ETL generalizes Transformation Based Learning by automatically solving the TBL bottleneck: the construction of good template sets. ETL uses the information gain in order to select the feature combinations that provide good template sets. We show that ETL can use the template evolution strategy to accelerate transformation learning. In this work, we also present ETL Committee, an ensemble method that uses ETL as the base learner.

We describe the application of ETL to four language independent NLP tasks: part-of-speech tagging, phrase chunking, named entity recognition and semantic role labeling. Overall, we successfully apply it to thirteen different corpora in six different languages: Dutch, English, German, Hindi, Portuguese and Spanish.

Our extensive experimental results demonstrate that ETL is an effective way to learn accurate transformation rules. In all experiments, ETL shows better results than TBL with hand-crafted templates. For the POS tagging task, ETL obtains state-of-the-art results for the four examined corpora: Mac-Morpho, Tycho Brahe, TIGER and Brown. For the PCK task, ETL shows state-of-the-art results for the SNR-CLIC Corpus, and state-of-the-art competitive results for the other three examined corpora: Ramshaw & Marcus, CoNLL-2000 and SPSAL-2007. For the NER task, ETL obtains state-of-the-art competitive results for the three examined corpora: HAREM, SPA CoNLL-2002 and DUT CoNLL-2002. For the SRL task, ETL obtains regular results for the two examined corpora: CoNLL-2004 and CoNLL-2005. We believe that by avoiding the use of handcrafted templates, ETL enables the use of transformation rules to a greater range of classification tasks.

We observe the following positive characteristics of the ETL approach:

1. The modeling phase is simple. It only requires a training set and a simple initial classifier. Moreover, using a common parameter setting, we achieve

competitive results for language independent POS tagging, PCK, NER and SRL tasks.

2. Simplifies the use of large feature sets, since it does not require hand-crafted templates. For instance, our ETL system for the SRL task uses 20 features.
3. Provides an effective way to handle high dimensional features. This property allows the use of the feature *word* in all experiments.
4. The *true class trick* allows the inclusion of the *current classification* feature in the generated templates. Hence, the TBL property of accessing intermediate results of the classification process is effectively used.
5. ETL training is reasonably fast. The observed ETL training time for the Mac-Morpho Corpus, using template evolution, is below one hour running on an Intel Centrino Duo 1.66GHz notebook. The template evolution strategy also accelerates transformation learning by a factor of five for the CoNLL-2000 Corpus.
6. The classifiers produced by ETL are fast. For instance, our Python implementation of the ETL PCK created with the CoNLL-2000 Corpus processes about 12,000 tokens per second.
7. Since the ETL inputs are just the training set and the initial classifier, it is easy to use it as a base learner for an ensemble method. This is demonstrated through the ETL Committee method.

Our experimental results also demonstrate that ETL Committee is an effective way to improve the ETL effectiveness. Using the ETL Committee approach, we obtain state-of-the-art competitive performance results in the thirteen corpus-driven tasks. For the PCK task, ETL Committee shows state-of-the-art results for the SPSAL-2007 Corpus. For the NER task, ETL Committee shows the best result reported so far for the HAREM Corpus.

We observe the following positive characteristics of the ETL Committee approach:

1. Improves the ETL effectiveness without human effort.
2. Particularly useful when dealing with very complex tasks that use large sets of features. For instance, we achieve state-of-the-art competitive performance for the SRL task.
3. The training and classification processes are very easy to parallelize, since each classifier is independent from the others.

Although ETL and ETL Committee approaches present good results for the examined tasks, they can be improved by exploring the issues that we enumerate next.

1. ETL templates are restricted to the combination of basic input features. On the other hand, TBL can process templates with *derived* features that are generated at training time. For instance, the rule

$$\text{pos}[0]=\text{DT} \quad \text{pos}[1,3]=\text{JJR} \quad \rightarrow \quad \text{post}=\text{RB}$$

uses the derived feature `pos[1,3]`. This feature checks the `pos` value of the three tokens following the current one. TBL derived features are very rich and complex [61], providing a powerful representation mechanism. Extending ETL to extract templates that use derived features is an open problem.

2. Creation of a template selection procedure. It is necessary for the cases where too many templates are generated, such as in SRL. A straightforward method is to use only the templates extracted up to a given tree level, exactly as we have done for the CoNLL-2005 Corpus. However, other entropy guided template selection methods could be tested.
3. To speed up the ETL Committee classification process. One possibility is to implement the Roche & Schabes [60] idea of converting transformation rules into deterministic finite-state transducers. According to Roche & Schabes, a transformation based English POS tagger converted into a finite-state tagger requires n steps to tag a sentence of length n , independently of the number of rules and the length of the context they require.
4. Investigation of other ensemble strategies.

Bibliography

- [1] V. Punyakanok, D. Roth and W. tau Yih. **The necessity of syntactic parsing for semantic role labeling**. In *In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1117–1123, 2005. 8.1
- [2] S. M. Aluísio, J. M. Pelizzoni, A. R. Marchi, L. de Oliveira, R. Manenti and V. Marquiasfável. **An account of the challenge of tagging a reference corpus for brazilian portuguese**. In *PROPOR*, pages 110–117, 2003. 5.1
- [3] C. N. Aranha. *Reconhecimento de entidades mencionadas em português*, chapter O Cortex e a sua participação no HAREM. Linguateca, Portugal, 2007. 7.4
- [4] A. PVS and K. Gali. **Part-of-speech tagging and chunking using conditional random fields and transformation based learning**. In *Proceedings of the IJCAI and the Workshop On Shallow Parsing for South Asian Languages (SPSAL)*, pages 21–24, 2007. 6.7
- [5] R. E. Banfield, L. O. Hall, K. W. Bowyer and W. P. Kegelmeyer. **Ensemble diversity measures and their application to thinning**. *Information Fusion*, 6(1):49–62, 2005. 2.5, 3.1(c)
- [6] **A comparison of decision tree ensemble creation techniques**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):173–180, 2007. Robert E. Banfield and Lawrence O. Hall and Kevin W. Bowyer and W. Philip Kegelmeyer. 3
- [7] A. Bharati and P. R. Mannem. **Introduction to shallow parsing contest on south asian languages**. In *Proceedings of the IJCAI and the Workshop On Shallow Parsing for South Asian Languages (SPSAL)*, pages 1–8, 2007. 6.1, 6.7
- [8] G. Biau, L. Devroye and G. Lugosi. **Consistency of random forests and other averaging classifiers**. *J. Mach. Learn. Res.*, 9:2015–2033, 2008. 3

- [9] T. Brants. **Tnt – a statistical part-of-speech tagger**. In *ANLP*, pages 224–231, 2000. 5.6
- [10] S. Brants, S. Dipper, S. Hansen, W. Lezius and G. Smith. **The TIGER treebank**. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, 2002. 5.1
- [11] L. Breiman. **Bagging predictors**. *Machine Learning*, 24(2):123–140, 1996. 3, 3.1(a)
- [12] L. Breiman. **Random forests**. *Mach. Learn.*, 45(1):5–32, 2001. 2.5, 3, 3.1(c), 3.3
- [13] E. Brill. **Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging**. *Comput. Linguistics*, 21(4):543–565, 1995. 1, 2.1, 5.2, 5.2(a), 5.3, 5.7, 7.3
- [14] G. Brown, J. L. Wyatt and P. Tiño. **Managing diversity in regression ensembles**. *Journal of Machine Learning Research*, 6:1621–1650, 2005. 3
- [15] S. Carberry, K. Vijay-Shanker, A. Wilson and K. Samuel. **Randomized rule selection in transformation-based learning: a comparative study**. *Natural Language Engineering*, 7(2):99–116, 2001. 2.7
- [16] X. Carreras, L. Màrques and L. Padró. **Named entity extraction using adaboost**. In *Proceedings of CoNLL-2002*, pages 167–170. Taipei, Taiwan, 2002. 7.5, 7.6
- [17] X. Carreras and L. Màrquez. **Introduction to the conll-2004 shared task: Semantic role labeling**. In H. T. Ng and E. Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 89–97, Boston, Massachusetts, USA, May 2004. Association for Computational Linguistics. 8, 8.1, 8.3
- [18] X. Carreras and L. Màrquez. **Introduction to the conll-2005 shared task: Semantic role labeling**. In *Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, MI, USA, 2005. Association for Computational Linguistics. 8.1, 8.1, 8.3
- [19] S. Corston-Oliver and M. Gamon. **Combining decision trees and transformation-based learning to correct transferred linguistic representations**. In *Proceedings of the Ninth Machine Translation Summit*, pages 55–62, New Orleans, USA, 2003. Association for Machine Translation in the Americas. 2.7

- [20] J. R. Curran and R. K. Wong. **Formalisation of transformation-based learning**. In *Proceedings of the ACSC*, pages 51–57, Canberra, Australia, 2000. 1, 2.4
- [21] M. Dash and H. Liu. **Feature selection for classification**. *Intelligent Data Analysis*, 1:131–156, 1997. 2.3(a)
- [22] T. G. Dietterich. **Ensemble methods in machine learning**. In *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15, London, UK, 2000. Springer-Verlag. 3
- [23] J. Elming. **Transformation-based corrections of rule-based mt**. In *Proceedings of the EAMT 11th Annual Conference*, Oslo, Norway, 2006. 2.2
- [24] M. Finger. **Técnicas de otimização da precisão empregadas no etiquetador tycho brahe**. In *Proceedings of PROPOR*, pages 141–154, São Paulo, November 2000. 1, 2.1
- [25] R. Florian, J. C. Henderson and G. Ngai. **Coaxing confidences from an old friend: Probabilistic classifications from transformation rule lists**. In *Proceedings of Joint Sigdat Conference on Empirical Methods in NLP and Very Large Corpora*, Hong Kong University of Science and Technology, October 2000. 2.1
- [26] R. Florian. **Named entity recognition as a house of cards: Classifier stacking**. In *Proceedings of CoNLL-2002*, pages 175–178. Taipei, Taiwan, 2002. 1, 2.1, 2.2
- [27] R. Florian. **Transformation based learning and data-driven lexical disambiguation: Syntactic and semantic ambiguity resolution**. PhD thesis, *The Johns Hopkins University*, 2002. 2.2, 2.6
- [28] R. Florian, A. Ittycheriah, H. Jing and T. Zhang. **Named entity recognition through classifier combination**. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 168–171. Edmonton, Canada, 2003. 3
- [29] G. Forman, I. Guyon and A. Elisseeff. **An extensive empirical study of feature selection metrics for text classification**. *Journal of Machine Learning Research*, 3:1289–1305, 2003. 2.3(a)
- [30] W. N. Francis and H. Kucera. **Frequency analysis of english usage**. *Lexicon and grammar*, 1982. 5.1

- [31] Y. Freund and R. E. Schapire. **A decision-theoretic generalization of on-line learning and an application to boosting.** *Journal of Computer and System Sciences*, 55(1):119–139, 1997. 3
- [32] M. C. Freitas, M. Garrao, C. Oliveira, C. N. dos Santos and M. Silveira. **A anotação de um corpus para o aprendizado supervisionado de um modelo de sn.** In *Proceedings of the III TIL / XXV Congresso da SBC*, São Leopoldo - RS - Brasil, 2005. 6.1
- [33] M. C. Freitas, J. C. Duarte, C. N. dos Santos, R. L. Milidiú, R. P. Renteria and V. Quental. **A machine learning approach to the identification of appositives.** In *Proceedings of Ibero-American AI Conference*, Ribeirão Preto, Brazil, October 2006. 1, 2.1
- [34] K. Hacioglu, S. S. Pradhan, W. H. Ward, J. H. Martin and D. Jurafsky. **Semantic role labeling by tagging syntactic chunks.** In H. T. Ng and E. Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, Boston, Massachusetts, USA, May 2004. Association for Computational Linguistics. 8.2(a), 8.4
- [35] D. Higgins. **A transformation-based approach to argument labeling.** In H. T. Ng and E. Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 114–117, Boston, Massachusetts, USA, May 6 - May 7 2004. Association for Computational Linguistics. 1, 2.1, 2.2, 8.3, 8.6
- [36] T. K. Ho. **The random subspace method for constructing decision forests.** *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):832–844, 1998. 2.5, 3, 3.1(c)
- [37] D. Jurafsky and J. H. Martin. **Speech and language processing.** Prentice Hall, 2000. 1, 5
- [38] T. Kudo and Y. Matsumoto. **Chunking with support vector machines.** In *Proceedings of the NAACL-2001*, 2001. 2.3(d), 6.5
- [39] F. Liu, Q. Shi and J. Tao. **Tree-guided transformation-based homograph disambiguation in mandarin tts system.** In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4657–4660, Cambridge, MA, USA, 2008. 2.7

- [40] L. Mangu and E. Brill. **Automatic rule acquisition for spelling correction.** In *Proceedings of The Fourteenth ICML*. Morgan Kaufmann, 1997. 2.1
- [41] M. P. Marcus, M. A. Marcinkiewicz and B. Santorini. **Building a large annotated corpus of english: the penn treebank.** *Computational Linguistics*, 19(2):313–330, 1993. 8.1
- [42] L. Màrquez, P. R. Comas, J. Giménez and N. Català. **Semantic role labeling as sequential tagging.** In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CONLL'05)*, pages 193–196, 2005. 8.2(a)
- [43] B. Megyesi. **Shallow parsing with pos taggers and linguistic features.** *Journal of Machine Learning Research*, 2:639–668, 2002. 1, 2.1
- [44] R. L. Milidiú, J. C. Duarte and R. Cavalcante. **Machine learning algorithms for portuguese named entity recognition.** In *Proceedings of Fourth Workshop in Information and Human Language Technology*, Ribeirão Preto, Brazil, 2006. 1, 2.1, 7.3, 7.4
- [45] R. L. Milidiú, C. N. dos Santos, J. C. Duarte and R. P. Renteria. **Semi-supervised learning for portuguese noun phrase extraction.** In *Proceedings of 7th Workshop on Computational Processing of Written and Spoken Portuguese*, pages 200–203, Itatiaia, Brazil, May 2006. 1, 2.1
- [46] R. L. Milidiú, J. C. Duarte and C. N. dos Santos. **Tbl template selection: An evolutionary approach.** In *Proceedings of Conference of the Spanish Association for Artificial Intelligence - CAEPIA*, Salamanca, Spain, 2007. 1, 2.7
- [47] R. L. Milidiú, J. C. Duarte and C. N. dos Santos. **Evolutionary tbl template generation.** *Journal of the Brazilian Computer Society*, 13(4):39–50, 2007. 1, 2.7
- [48] R. L. Milidiú, C. N. dos Santos and J. C. Duarte. **Portuguese corpus-based learning using etl.** *Journal of the Brazilian Computer Society*, 14(4):17–27, 2008. 1, 2
- [49] R. L. Milidiú, C. N. dos Santos and J. C. Duarte. **Phrase chunking using entropy guided transformation learning.** In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - ACL-08: HLT*, Columbus, Ohio, 2008. 1, 2, 5.1

- [50] T. M. Mitchell. **Machine learning**. McGraw-Hill, New York, 1997. 2.3(a), 2.3(e)
- [51] G. Ngai and R. Florian. **Transformation-based learning in the fast lane**. In *Proceedings of North American ACL*, pages 40–47, June 2001. 2.4, 4.5, 5.4
- [52] N. C. Oza and K. Tumer. **Classifier ensembles: Select real-world applications**. *Information Fusion*, 9(1):4–20, 2008. 3
- [53] M. Palmer, D. Gildea and P. Kingsbury. **The proposition bank: An annotated corpus of semantic roles**. *Comput. Linguist.*, 31(1):71–106, 2005. 8.1
- [54] P. Panov and S. Dzeroski. **Combining bagging and random subspaces to create better ensembles**. In *7th International Symposium on Intelligent Data Analysis*, pages 118–129, Ljubljana, Slovenia, 2007. 3, 3.3
- [55] N. García-Pedrajas and D. Ortiz-Boyer. **Boosting random subspace method**. *Neural Networks*, 21(9):1344–1362, 2008. 3
- [56] J. R. Quinlan. **Induction of decision trees**. *Machine Learning*, 1(1):81–106, 1986. 2.3(a), 2.3(b)
- [57] J. R. Quinlan. **C4.5: programs for machine learning**. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. 2.3, 2.3(b), 4.5
- [58] L. Ramshaw and M. Marcus. **Exploring the statistical derivation of transformational rule sequences for part-of-speech tagging**. In *Proceedings of the Balancing Act - Workshop on Combining Symbolic and Statistical Approaches to Language*, pages 86–95. Association for Computational Linguistics, 1994. 2.2
- [59] L. Ramshaw and M. Marcus. **Text chunking using transformation-based learning**. In S. Armstrong, K. Church, P. Isabelle, S. Manzi, E. Tzoukermann and D. Yarowsky, editors, *Natural Language Processing Using Very Large Corpora*. Kluwer, 1999. 1, 2.1, 2.2, 6.1, 6.3
- [60] E. Roche and Y. Schabes. **Deterministic part-of-speech tagging with finite-state transducers**. *Comput. Linguist.*, 21(2):227–253, 1995. 3
- [61] C. N. dos Santos and C. Oliveira. **Constrained atomic term: Widening the reach of rule templates in transformation based learning**. In *Portuguese Conference on Artificial Intelligence - EPIA*, pages 622–633, 2005. 1, 2.1, 2.2, 6.3, 6.4, 1

- [62] C. N. dos Santos and R. L. Milidiú. **Entropy guided transformation learning**. Technical Report 29/07, Departamento de Informática, PUC-Rio, 2007. 1, 2, 2.7
- [63] C. N. dos Santos and R. L. Milidiú. **Probabilistic classifications with tbl**. In *Proceedings of Eighth International Conference on Intelligent Text Processing and Computational Linguistics – CICLing*, pages 196–207, Mexico City, Mexico, February 2007. 1, 2.1
- [64] D. Santos and N. Cardoso. **Reconhecimento de entidades mencionadas em português**. Linguateca, Portugal, 2007. 7.1, 7.1, 7.3, 7.4
- [65] C. N. dos Santos, R. L. Milidiú and R. P. Rentería. **Portuguese part-of-speech tagging using entropy guided transformation learning**. In *Proceedings of 8th Workshop on Computational Processing of Written and Spoken Portuguese*, pages 143–152, 2008. 1, 1, 2, 2.1, 5.4, 5.5
- [66] C. N. dos Santos and R. L. Milidiú. *Learning and Approximation: Theoretical Foundations and Applications*, volume 1 of *Foundations of Computational Intelligence*, chapter Entropy Guided Transformation Learning. Springer, 2009. 1, 2
- [67] L. Sarmiento, A. Sofia and L. Cabral. **Repentino - a wide-scope gazetteer for entity recognition in portuguese**. In *Proceedings of 7th Workshop on Computational Processing of Written and Spoken Portuguese*, pages 31–40, Itatiaia, Brazil, May 2006. 7.3
- [68] W. Skut, B. Krenn, T. Brants and H. Uszkoreit. **An annotation scheme for free word order languages**. In *In Proceedings of ANLP-97*, 1997. 5.6
- [69] J. Su and H. Zhang. **A fast decision tree learning algorithm**. In *AAAI*, 2006. 2.3
- [70] M. Surdeanu, L. Màrquez, X. Carreras and P. Comas. **Combination strategies for semantic role labeling**. *Journal of Artificial Intelligence Research (JAIR)*, 29:105–151, 2007. 1, 8, 8.1, 8.5
- [71] M. Surdeanu, R. Johansson, A. Meyers, L. Màrquez and J. Nivre. **The conll 2008 shared task on joint parsing of syntactic and semantic dependencies**. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England, August 2008. Coling 2008 Organizing Committee. 1, 2.3(d)

- [72] IEL-UNICAMP and IME-USP. **Corpus anotado do português histórico tycho brahe**. <http://www.ime.usp.br/~tycho/corpus/>. accessed in january 23, 2008. 5.1
- [73] E. F. T. K. Sang and S. Buchholz. **Introduction to the conll-2000 shared task: chunking**. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th CONLL*, pages 127–132, Morristown, NJ, USA, 2000. Association for Computational Linguistics. 1, 6, 6.1
- [74] E. F. Tjong Kim Sang. **Introduction to the conll-2002 shared task: Language-independent named entity recognition**. In *Proceedings of CoNLL-2002*, pages 155–158. Taipei, Taiwan, 2002. 1, 7.1, 7.1, 7.3
- [75] E. F. Tjong Kim Sang and F. De Meulder. **Introduction to the conll-2003 shared task: Language-independent named entity recognition**. In W. Daelemans and M. Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada, 2003. 1
- [76] Y.-C. Wu, C.-H. Chang and Y.-S. Lee. **A general and multi-lingual phrase chunking model based on masking method**. In *Proceedings of 7th International Conference on Intelligent Text Processing and Computational Linguistics*, pages 144–155, 2006. 6.6
- [77] Y.-S. Hwang, H.-J. Chung and H.-C. Rim. **Weighted probabilistic sum model based on decision tree decomposition for text chunking**. *International Journal of Computer Processing of Oriental Languages*, (1):1–20, March 2003. 2.7

A

ETL Committee Behavior

This appendix presents some results on the behavior of the ETL Committee learning strategy. We use the SRL CoNLL-2004 Corpus in all the experiments shown in this appendix. This corpus is described in section 8.1. The development corpus is used to assess the system results.

This appendix is organized as follows. In section A.1, we investigate the behavior of the ETL Committee performance as the ensemble size increases. In section A.2, we analyze the ensemble performance sensitivity to the percentage of sampled features. In section A.3, we investigate the contribution of redundant rules. In section A.4, we examine the variance on the classification results due to the random nature of ETL Committee.

A.1 Ensemble Size

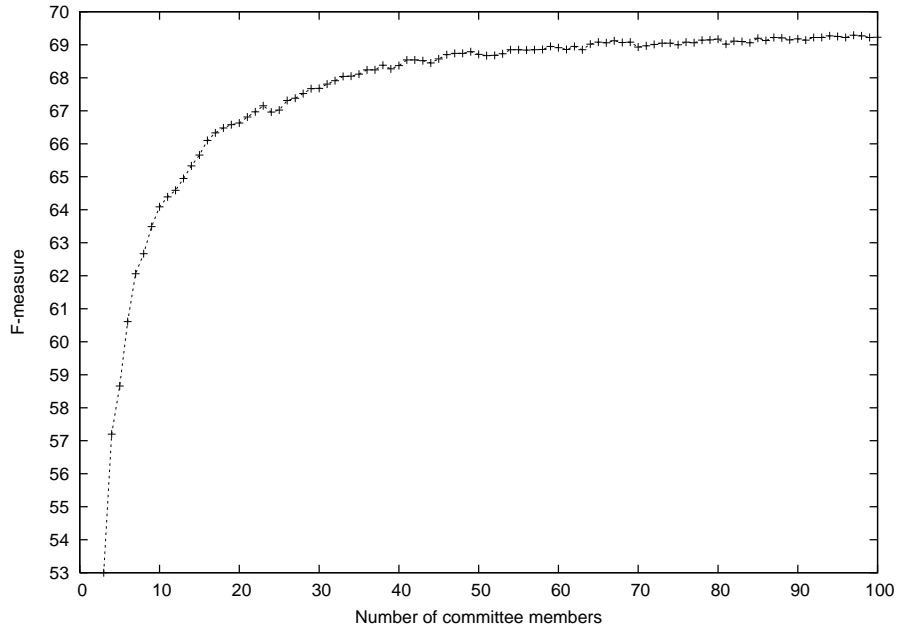
The *ensemble size* is defined by the number of bootstrap samplings. Here, we examine how the ETL Committee performance behave as the number of committee members increases.

Figure A.1 demonstrates the relationship of the $F_{\beta=1}$ for a given number of ETL classifiers in the ensemble. We can see that the ensemble performance increases rapidly until approximately 40 classifiers are included. Then, the $F_{\beta=1}$ increases slowly until it gets stable with around 100 classifiers. Note that using just 50 models we have a $F_{\beta=1}$ of 68.7. ETL Committee has a similar behavior in the other three tasks: POS tagging, PCK and NER.

A.2 Feature Sampling

The *feature sampling parameter* indicates the percentage of input features to be used in the training of each committee member. In this section, we investigate the ensemble performance sensitivity to the percentage of sampled features.

In Table A.1, we show the ETL Committee performance for different values of the feature sampling parameter. For this experiment, we create

Figure A.1: $F_{\beta=1}$ x Number of committee members curve.

ensembles of 50 classifiers. The best performance occurs when 70% of the features are randomly sampled for each classifier. In this case, the $F_{\beta=1}$ increases by about 0.7 when compared to the result in the first table line, where all features are used. In Table A.1, we can see that even using only 50% of the features, the performance does not degrade. However, using less than 70% of the features can lead to poor results for tasks with a few number of features such as POS tagging and PCK.

Table A.1: ETL Committee performance sensitivity to the percentage of sampled features.

Percentage of sampled features	Precision (%)	Recall (%)	$F_{\beta=1}$
100%	75.43	61.95	68.03
90%	75.97	62.21	68.40
80%	76.40	62.32	68.65
70%	76.44	62.40	68.71
60%	76.52	61.87	68.42
50%	76.64	61.50	68.24

A.3 Redundant Rules

In this section, we analyze the contribution of redundant rules for the ETL Committee performance. In Table A.2, we show the ETL Committee performance when redundant rules are used or not used. For this experiment,

we create ensembles of 50. The result in the first table line corresponds to the default ETL Committee method, which uses redundant rules. The second table line presents the ensemble performance when redundant rules are not used. In this case, the $F_{\beta=1}$ drops by about two points. This finding indicates that the overfitting provided by redundant rules is very important to the construction of more diverse ETL classifiers. ETL Committee has a similar behavior in the other three tasks.

Table A.2: Contribution of redundant rules for the ETL Committee performance.

Redundant rules	Precision (%)	Recall (%)	$F_{\beta=1}$
YES	76.44	62.40	68.71
NO	76.63	59.10	66.73

A.4 ETL Committee Random Nature

ETL Committee uses randomized methods to build diverse classifiers. Due to this random nature, the performance of two different ETL Committee models may vary, even if they are trained using exactly the same parameter setting. In this section, we investigate how the ETL Committee performance varies with different algorithm runs. We perform an experiment that consists in training ten different committees using the same parameter setting. Each committee contains 50 classifiers. We use the SRL CoNLL-2004 development set to assess the performance of each ensemble model.

Table A.3: Results of ten different ETL Committee runs.

Run	Precision (%)	Recall (%)	$F_{\beta=1}$
1	75.80	62.14	68.29
2	76.02	62.03	68.31
3	75.97	62.21	68.40
4	76.18	62.13	68.44
5	76.33	62.17	68.53
6	76.34	62.17	68.53
7	76.30	62.23	68.55
8	76.47	62.15	68.57
9	76.27	62.31	68.59
10	76.66	62.12	68.63

In Table A.3, we show the ETL Committee performance for the ten different runs. The standard deviation of the F-score is 0.12. Since this value

is very small, it can be concluded that the random nature of the learning algorithm has a little impact in the variance of the final result. The standard deviation is even smaller if we use a larger number of classifiers in the committee. In this experiment we use 50 classifiers. However, we use ensembles of size 100 for the experiments shown in chapters 5, 6, 7 and 8. Therefore, the reported ETL Committee results are stable.