Clause Identification using Entropy Guided Transformation Learning

Eraldo R. Fernandes[1,2], **Bernardo A. Pires**[1],
Cícero N. dos Santos[1,3], **Ruy L. Milidiú**[1]

[1] Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro, Brazil

[2]Laboratório de Automação
Instituto Federal de Educação, Ciência e Tecnologia de Goiás (IFG)
Jataí, Brazil

[3]Mestrado em Informática Aplicada - MIA
Universidade de Fortaleza - UNIFOR
Fortaleza, Brazil

{efernandes,bpires,milidiu}@inf.puc-rio.br, cnogueira@unifor.br

Abstract. *Entropy Guided Transformation Learning (ETL) is a machine learning strategy that extends Transformation Based Learning by providing automatic template generation. In this work, we propose an ETL approach to the clause identification task. We use the English language corpus of the CoNLL'2001 shared task. The achieved performance is not competitive yet, since the* $F_{\beta=1}$ *of the ETL based system is* 80.55*, whereas the state-of-the-art system performance is* 85.03*. Nevertheless, our modeling strategy is very simple, when compared to the state-of-the-art approaches. These first findings indicate that the ETL approach is a promising one for this task. One can enhance its performance by incorporating problem specific knowledge. Additional features can be easily introduced in the ETL model.*

1. Introduction

Clause identification is a Natural Language Processing task consisting of splitting a sentence into clauses. A clause is defined as a word sequence which contains a subject and a predicate. Clause identification is a special kind of shallow parsing, like text chunking [Milidiú et al. 2008]. Nevertheless, it is more difficult than text chunking, since clauses can have embedded clauses. Clause information is important for several more elaborated tasks such as full parsing and semantic role labeling.

The state-of-the-art system for clause identification in English texts [Carreras et al. 2005] is based on an elaborated method and shows a $F_{\beta=1}$ value of 85.03. It uses perceptrons to filter and rank candidate clauses and dynamic programming to optimize over the candidates combination. When classifying clauses with embedded clauses, the method explores some features that inform about the internal clause structures. Some other approaches have been successfully applied to this task. Boosting-trees-centered models [Carreras et al. 2002, Carreras and Màrquez 2001] show, respectively, the second and third best results for this task.

In this work, we propose an *Entropy Guided Transformation Learning* (ETL) system for clause identification. ETL [dos Santos and Milidiú 2009] is a machine learning strategy that generalizes Transformation Based Learning (TBL) [Brill 1995] by automatically solving the TBL bottleneck: the construction of good template sets. ETL uses the Information Gain measure in order to select the feature combinations that provide good template sets. First, ETL employs decision tree induction to perform an entropy guided template generation. Next, it applies the TBL algorithm to learn a set of transformation rules. ETL is an effective way to eliminate the need of a problem domain expert to build TBL templates.

Since our approach is corpus-based, we use the English language corpus of the CoNLL'2001 shared task [Sang and Déjean 2001]. In Table 1, we show the best result obtained with our system. The achieved performance is not competitive yet, since the $F_{\beta=1}$ of the ETL based system is 80.55, whereas the state-of-the-art system performance is 85.03. Nevertheless, our modeling strategy is very simple, when compared to the state-of-the-art approaches. These first findings indicate that the ETL approach is a promising one for this task. One can enhance its performance by incorporating problem specific knowledge. Additional features can be easily introduced in the ETL model.

Table 1. Clause identification performances on CoNLL'2001 corpus.

System	*Precision*	*Recall*	$F_{\beta=1}$
State-of-the-art	88.17	81.10	85.03
CoNLL'2001	84.82	78.85	81.73
ETL	86.37	75.45	80.55

The remaining of this paper is structured as follows. In Section 2, we describe the ETL method. In Section 3, we present the ETL modeling for clause identification. In Section 4, the experimental results are reported and discussed. Finally, in Section 5, we present our concluding remarks.

2. Entropy Guided Transformation Learning

Entropy Guided Transformation Learning [Milidiú et al. 2008] generalizes Transformation Based Learning by automatically generating rule templates. ETL employs an *entropy guided template generation* approach, which uses *Information Gain* in order to select the feature combinations that provide good template sets. ETL has been successfully applied to part-of-speech tagging [dos Santos et al. 2008], phrase chunking, and named entity recognition [dos Santos and Milidiú 2009, Milidiú et al. 2008], producing results at least as good as the ones of TBL with handcrafted templates. Several ETL-based language processors, for different languages, are freely available on the Web through the F-EXT-WS[1] service [Fernandes et al. 2009]. The ETL algorithm is illustrated in Figure 1 and briefly reviewed in the next two subsections. A detailed description of ETL can be found in [Milidiú et al. 2008, dos Santos and Milidiú 2009].

2.1. Entropy Guided Template Generation

Information gain, which is based on the data entropy, is a key strategy for feature selection. The most popular Decision Tree (DT) learning algorithms [Quinlan 1993,

[1] http://www.learn.inf.puc-rio.br/

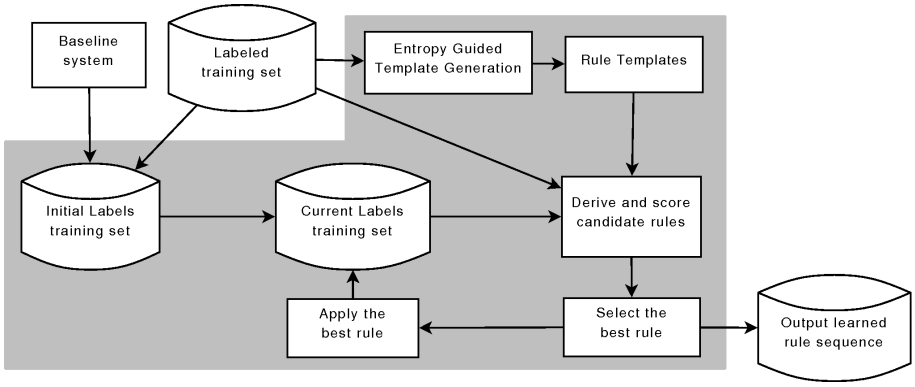

Figure 1. Entropy Guided Transformation Learning.

Su and Zhang 2006] implement this strategy. 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.

ETL uses a very simple DT decomposition scheme for template generation. The decomposition process includes 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.

2.2. Transformation Based Learning

In the ETL approach, after the template set is generated, the learning of transformation rules is done by using the TBL algorithm. The TBL algorithm can be formulated as follows:

1. The baseline system (BLS) is applied to the training set, in order to obtain an initial classification;
2. REPEAT
   (a) The resulting classification is compared with the correct one and, whenever a classification error is found, all the rules that can correct it are generated by instantiating the templates. Usually, a new rule will correct some errors, but will also generate some other errors by changing correctly classified samples;
   (b) The rules' scores are computed. This score is defined as the difference between the total number of repaired errors and the total number of created errors.
   (c) If there is no rule with a score above a given rule score threshold, STOP the learning process;
   (d) The best scoring rule is selected, stored in the set of learned rules and applied to the training set;

3. Clause Identification using ETL

In this section, we show our ETL modeling for the clause identification task. We use the CoNLL'2001 corpus, which was automatically generated from the Wall Street Journal part of the Penn Treebank [Marcus et al. 1993]. In Figure 2, we present an example of a sentence from the Penn Treebank divided into clauses by parentheses. The CoNLL'2001

corpus provides three input features for each token: word, POS tag, and chunk tag. These are not the Peen Treebank golden values. The POS and chunk tags are respectively generated using Brill's tagger [Brill 1994] and Sang's tagger [Sang 2000]. Hence, the performance results provide more realistic estimates for the expected behavior on unannotated datasets.

```
( Not everyone believes
  ( that
    ( the good times are over for shippers )
  )
  .
)
```

Figure 2. Sentence from Penn Treebank split into clauses.

We approach the clause identification problem in three steps, as generally adopted in the CoNLL'2001 shared task. The three steps are: (i) clause start identification; (ii) clause end identification; and (iii) complete clause identification. We solve these three problems sequentially. Therefore, we can use the information produced in previous steps as input to the next ones. First, we use *start* tags as input for the end classifier. Next, we use both *start* and *end* tags to identify the complete clauses.

3.1. Baseline System

We adopt the simple baseline system proposed in the CoNLL'2001 shared task. This BLS assigns just one clause for the whole sentence. This system is used in the three steps.

3.2. Clause Boundary Candidates

The first and second steps identify the clause boundary candidates, that is, start and end tokens. These steps identify the tokens that are good candidates to clause boundaries, without any concern to consistence among them. We model these two subtasks as token-classification problems. In Table 2, we illustrate the corpus format through an example. Observe that each line corresponds to a token. This example corresponds to the sentence in Figure 2. The *Start* and *End* columns in the table respectively indicate the *start* and *end* classifications. In the first step, if a token *starts* one or more clauses, it must be classified as *S*, otherwise, it must be classified as *X*. Similarly, in the second step, if a token *ends* one or more clauses, it must be classified as *E*, otherwise as *X*.

3.3. Complete Clause Identification

The last and most difficult step consists of splitting a given sentence into clauses. In the CoNLL corpus, the complete clauses within a sentence are encoded through a unique token feature using the following tags: *(S** – indicating that the token starts a clause; **S)* – indicating that the token ends a clause; * – representing a token that neither starts nor ends a clause; and any combinations of the previous to represent tokens that start or end more than one clause. The *Clause* column in Table 2 contains the tags that encode the clauses within the sentence as illustrated in Figure 2.

For the third step, we present two modeling approaches: *token classification* and *pair classification*. These modeling approaches are detailed in the following subsections.

Table 2. CoNLL'2001 corpus format.

Word	*Start*	*End*	*Clause*
Not	S	X	(S*
everyone	X	X	*
believes	X	X	*
that	S	X	(S*
the	S	X	(S*
good	X	X	*
times	X	X	*
are	X	X	*
over	X	X	*
for	X	X	*
shippers	X	E	*S)S)
.	X	E	*S)

3.3.1. ETL-Token

Using a token classification approach, we perform the third step in a straightforward manner with ETL. We train an ETL model to classify each token as *, *(S**, **S)*, or any tag combination appearing in the *Clause* column of the training corpus. This approach is very simple but also limited. We observe that many clauses are tokenwise long. For instance, in the CoNLL training corpus, the fraction of clauses with length longer than 14 tokens is greater than 40%. Even using a window of 27 tokens (the current token plus the thirteen tokens on each side), when classifying one clause boundary the other one is not included for 40% of the clauses. We observe that this window size is computationally prohibitive for the ETL algorithm.

3.3.2. ETL-Pair

In order to capture more clause context information, we try a second modeling approach. This approach uses the output of the start and end classifiers to create a new corpus. For each start-end pair of tokens from a given sentence in the original corpus, we generate one example in the new corpus. Next, we train an ETL model that learns to classify which pairs of tokens define clause boundaries.

3.4. Derived Features

We use the three input features provided in the CoNLL corpus plus some derived features. We derive these additional features in the same fashion as in [Carreras et al. 2002], although we use just a small subset of the features proposed by these authors.

The selected features inform us about the occurence of relevant elements within a sentence fragment. The following elements are the relevant ones: *verb chunks*, *start tokens*, and *end tokens*. We call *verb chunks* the ones with chunk tag with value `verb`. We generate two features for each relevant element: a flag indicating its occurence; and the number of its occurences within a sentence fragment.

For the *Start*, *End*, and *ETL-Token* classifiers we use the same feature derivation scheme. For each token we derive twelve features: six for the sentence fragment before the token; and six for the sentence fragment after it. For the *ETL-Pair* classifier we use a different scheme. For each start-end pair of tokens we derive eighteen features: six for the sentence fragment before the start token; six for the sentence fragment after the end token; and six for the sentence fragment between the start-end tokens. Observe that a derived feature is only used when its required information is available.

4. Experiments

We use the corpus provided for the CoNLL'2001 shared task. This corpus was generated from the Wall Street Journal part of the Penn Treebank [Marcus et al. 1993] and is divided into three parts: (i) *train* – containing 8,936 sentences from sections 15 to 18; (ii) *development* – containing 2,012 sentences from section 20; and (iii) *test* – containing 1,671 sentences from section 21.

We use the development dataset for parameter tunning. For both the *Start* and *End* classifiers, we set the window size parameter to 5 and the rule score threshold to 2. For the *ETL-Token* classifier we set the window size to 7 and the rule threshold to 4; whereas for the *ETL-Pair* classifier the values 9 and 4 are respectively used.

In Table 3, we present the results for the *Start* and *End* classifiers in the development and test datasets. In this table we also report a comparison between the *ETL-Token* and *ETL-Pair* classifiers, since they are two alternatives to the same classification problem. The $F_{\beta=1}$ of the *ETL-Pair* system shows more than 4 points than the one of the *ETL-Token* system in both development and test datasets. We believe this improvement is due to *ETL-Pair* use of stronger information about the clause candidates.

Table 3. Results obtained for the ETL classifiers in the development and test datasets.

Strategy	*Development*			*Test*		
	Precision	*Recall*	$F_{\beta=1}$	*Precision*	*Recall*	$F_{\beta=1}$
Start	94.02	90.86	92.42	92.16	87.81	89.93
End	89.00	88.98	88.99	88.71	88.56	88.63
ETL-Token	82.55	75.88	79.07	80.47	72.28	76.16
ETL-Pair	87.67	78.98	83.10	86.37	75.45	80.55

In Table 4, we rank four competing classifiers: the state-of-the-art system [Carreras et al. 2005]; the second best system [Carreras et al. 2002]; the best system in the CoNLL'2001 shared task [Carreras and Màrquez 2001]; and the *ETL-Pair* system. Additionally, we report the performance of the BLS used by our ETL classifiers. Our results so far are getting competitive with the best clause identifiers.

5. Conclusions

In this paper, we present a machine learning based system to the clause identification problem. Our system uses the machine learning technique called Entropy Guided Transformation Learning (ETL). We use the English language corpus provided for the CoNLL'2001 shared task. The problem is divided into three steps: (i) clause start identification; (ii)

Table 4. Comparison with state-of-the-art results in the test dataset.

Strategy	*Precision*	*Recall*	$F_{\beta=1}$
[Carreras et al. 2005]	88.17	81.10	85.03
[Carreras et al. 2002]	90.18	78.11	83.71
[Carreras and Màrquez 2001]	84.82	78.85	81.73
ETL-Pair	86.37	75.45	80.55
BLS	98.44	33.88	50.41

clause end identification; and (iii) complete clause identification. We report the performance of the proposed system for the three steps. Our modeling strategy is very simple, when compared to the state-of-the-art approaches. These first findings indicate that the ETL approach is a promising one for this task.

We plan to improve the start and end classifiers. The state-of-the-art approach [Carreras et al. 2005] explores some features that inform about the structure of the embedded clauses, when classifying compound clauses. This idea can be introduced in our approach to improve its perfomance. We plan also to build ensemble models as in [Carreras et al. 2005], using ETL as the weak learner.

The Floresta Sintá(c)tica project [Freitas et al. 2008] provides a treebank for several Portuguese language texts. The Bosque part of the Floresta Sintá(c)tica corpus has been mannually reviewed and can be used for development of supervised machine learning approaches. We have generated a Portuguese clause corpus, following the CoNLL'2001 format, from the Bosque corpus. We are currently working on a clause identification system using this corpus, achieving some promising preliminary results.

References

Brill, E. (1994). Some advances in transformation-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 722–727, Seattle, USA.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Carreras, X. and Màrquez, L. (2001). Boosting trees for clause splitting. In *Proceedings of Fifth Conference on Computational Natural Language Learning*, Toulouse, France.

Carreras, X., Màrquez, L., and Castro, J. (2005). Filtering-ranking perceptron learning for partial parsing. *Machine Learning*, 60(1–3):41–71.

Carreras, X., Màrquez, L., Punyakanok, V., and Roth, D. (2002). Learning and inference for clause identification. In *Proceedings of the Thirteenth European Conference on Machine Learning*, pages 35–47.

dos Santos, C. N. and Milidiú, R. L. (2009). *Foundations of Computational Intelligence, Volume 1: Learning and Approximation*, volume 201 of *Studies in Computational Intelligence*, chapter Entropy Guided Transformation Learning, pages 159–184. Springer.

dos Santos, C. N., Milidiú, R. L., and Renteria, R. P. (2008). Portuguese part-of-speech tagging using entropy guided transformation learning. In *Proceedings of PROPOR 2008*, Aveiro, Portugal.

Fernandes, E. R., dos Santos, C. N., and Milidiú, R. L. (2009). Portuguese language processing service. In *Proceedings of the Web in Ibero-America Alternate Track of the 18th World Wide Web Conference*, Madrid.

Freitas, C., Rocha, P., and Bick, E. (2008). Floresta Sintá(c)tica: Bigger, thicker and easier. In Teixeira, A., de Lima, V. L. S., de Oliveira, L. C., and Quaresma, P., editors, *Computational Processing of the Portuguese Language*, volume 5190 of *Lecture Notes in Computer Science*, pages 216–219. Springer.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Milidiú, R. L., dos Santos, C. N., and Duarte, J. C. (2008). Phrase chunking using entropy guided transformation learning. In *Proceedings of ACL-08: HLT*, pages 647–655, Columbus, USA. Association for Computational Linguistics.

Milidiú, R. L., dos Santos, C. N., and Duarte, J. C. (2008). Portuguese corpus-based learning using ETL. *Journal of the Brazilian Computer Society*, 14(4).

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, USA.

Sang, E. F. T. K. (2000). Text chunking by system combination. In *Proceedings of Conference on Computational Natural Language Learning*, Lisbon, Portugal.

Sang, E. F. T. K. and Déjean, H. (2001). Introduction to the conll-2001 shared task: Clause identification. In *Proceedings of Fifth Conference on Computational Natural Language Learning*, Toulouse, France.

Su, J. and Zhang, H. (2006). A fast decision tree learning algorithm. In *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence*.