J Braz Comput Soc (2010) 16: 191–199
DOI 10.1007/s13173-010-0018-y

ORIGINAL PAPER

RelHunter: a machine learning method for relation extraction from text

Eraldo R. Fernandes · Ruy L. Milidiú · Raúl P. Rentería

Received: 8 February 2010 / Accepted: 16 July 2010 / Published online: 8 August 2010

Abstract We propose *RelHunter*, a machine learning-based method for the extraction of structured information from text. *RelHunter*'s key idea is to model the target structures as a relation over entities. Hence, the modeling effort is reduced to the identification of entities and the generation of a candidate relation, which are simpler problems than the original one. *RelHunter* fits a very broad spectrum of complex computational linguistic problems. We apply it to five tasks: phrase chunking, clause identification, hedge detection, quotation extraction, and dependency parsing. We compare *RelHunter* to token classification approaches through several computational experiments on seven multilingual corpora. *RelHunter* outperforms the token classification approaches by 2.14% on average. Moreover, we compare the derived systems against state-of-the-art systems for each corpus. Our systems achieve state-of-the-art performances for three corpora: Portuguese phrase chunking, Portuguese clause identification, and English quotation extraction. Additionally, the derived systems show good quality performance for the other four corpora.

This work was partially funded by CNPq and FAPERJ grants 557.128/2009-9 and E-26/170028/2008. The first author holds a CNPq doctoral fellowship and is supported by Instituto Federal de Educação, Ciência e Tecnologia de Goiás, Brazil.

E.R. Fernandes (✉) · R.L. Milidiú
Departamento de Informática, PUC-Rio, R. Mq. de São Vicente 225, Rio de Janeiro, 22.451-900, Brasil
e-mail: efernandes@inf.puc-rio.br

R.L. Milidiú
e-mail: milidiu@inf.puc-rio.br

R.P. Rentería
Microsoft Enterprise Search Group, Redmond, USA

Keywords Natural language processing · Entity relation extraction · Machine learning · Entropy Guided Transformation Learning

1 Introduction

Over the last two decades, several computational linguistic problems have been modeled as local token classification tasks and successfully approached by machine learning (ML) methods [1, 18]. Nevertheless, the harder problems consist in identifying complex structures within a text. These structures comprise many tokens and show nonlocal token dependencies.

Phrase Chunking [23] is a task that involves structure recognition. Punyakanok and Roth [21] decompose this task into three ML subtasks and combine their outputs using a dynamic programming algorithm. They show that this approach significantly outperforms a simple token classification model.

Clause Identification [24] is another computational linguistic task that requires structure recognition. As clauses may embed other clauses, these structures involve stronger dependencies than phrase chunks. Carreras et al. [3] propose an approach to clause identification that extends Punyakanok and Roth's previous work.

Phrase Recognition is a general task that includes both Phrase Chunking and Clause Identification. Carreras et al. [4] propose the Filtering-Ranking Perceptron (FRP) system for this general task. The FRP task modeling is strongly related to previous proposals [3, 21]; however, FRP *simultaneously* learns to solve the three subtasks. FRP is very effective although computationally expensive in terms of both training and prediction time.

Here, we describe *RelHunter*, a new ML-based method for the extraction of structured information from text. The

Table 1 Information extraction performances—F_1 measure

Language	Task	*RelHunter*	State of the Art
Portuguese	Chunking	87.08	87.08
	Clause	70.14	70.14
	Dependency	88.18	91.36
English	Chunking	92.26	94.12
	Clause	80.68	85.03
	Hedge	54.05	57.32
	Quotation	89.67	89.67

central idea of *RelHunter* is to model the target structures as a relation over entities. To learn how to extract the entities and the relation, the method uses two additional schemes: task decomposition and interdependent classification. The *RelHunter*'s task decomposition strategy is inspired by the modeling in [21] for Phrase Chunking, in [3] for Clause Chunking, and in [4] for Phrase Recognition. *RelHunter* uses Entropy Guided Transformation Learning (ETL) [6, 16] as its basic learning engine. ETL is an ML algorithm that performs interdependent classification in an effort-free modeling way.

RelHunter fits a broad spectrum of complex computational linguistic problems. To illustrate *RelHunter*'s generality, we apply it to five tasks: Phrase Chunking [23], Clause Identification [24], Hedge Detection [8], Quotation Extraction [5], and Dependency Parsing [19]. To evaluate the performance of the proposed system, we perform several experiments using seven annotated corpora, where three contain Portuguese text and four contain English text. In Table 1, we summarize the *RelHunter*'s performances along with the state-of-the-art performances on the evaluated corpora.

Preliminary versions of this method have been published in [9, 11, 12]. The remainder of this text is organized as follows. In Sect. 2, we discuss some approaches related to this work. In Sect. 3, we give an overview of the *RelHunter* method. We describe the task decomposition strategy in Sect. 4. Next, in Sect. 5, we detail the ETL algorithm and the *RelHunter*'s interdependent strategy. In Sect. 6, we present the modeling approaches for five natural language processing tasks. In Sect. 7, we report our empirical setup and findings on seven corpora. Finally, in Sect. 8, we present our conclusions and remarks.

2 Related work

Phrase Chunking [23] is an important computational linguistic task that involves structure recognition and consists in identifying nonoverlapping chunks of syntactically correlated words. Punyakanok and Roth [21] divide this problem into three machine learning subtasks. In the first subtask, they train a token classification model to identify tokens that start a chunk. Similarly, in the second subtask, they train a model to identify tokens that end a chunk. Next, a third model scores the chunk boundary candidates, i.e., all the pairs of start and end tokens. Finally, a dynamic programming algorithm generates a chunk set that maximizes the sum of the scoring function and satisfies the chunk formation constraints. They use hidden Markov models to train the three models and show that this approach significantly outperforms a simple token classification approach.

[They say [the good times are over for shippers] .]

Fig. 1 An English sentence and its clauses between brackets

Clause Identification [24] is another important computational linguistic task that consists in identifying the clauses within a sentence. A clause may embed other clauses, and thus these structures involve stronger dependencies than phrase chunks. Carreras et al. [3] propose an extension of Punyakanok and Roth's work for clause identification. Their system comprises complex methods for training and extraction in order to exploit the specific dependency aspects of clause structures.

Carreras et al. [4] propose the Filtering-Ranking Perceptron (FRP) system for two phrase recognition problems: phrase chunking and clause identification. The FRP modeling is strongly related to the two previously cited systems. Thus, it divides the original problem into three ML subtasks (start, end, and score) and combines the outputs using a dynamic programming algorithm. However, the training algorithm is a modified perceptron that simultaneously trains the three subtask models. In each perceptron iteration, the misidentified phrases are used to adjust the parameters of the three ML models. Hence, the training is guided by a global performance measure. FRP is very effective, although computationally expensive in terms of both training and prediction time. It is currently the state of the art for clause identification.

3 RelHunter overview

In this section, we present an overview of the *RelHunter* method. We use the Clause Identification task as an illustrative example. In Fig. 1, we show a sentence along with its two clauses indicated by brackets.

RelHunter's key idea is to model the target structures as a relation over entities. To learn how to extract the entities and the relation, the method relies on two additional schemes: *task decomposition* and *interdependent classification*.

RelHunter decomposes the original task into three subtasks: (i) *Entity Identification*; (ii) *Candidate Relation Generation*; and (iii) *Relation Recognition*. In Fig. 2, we illus-

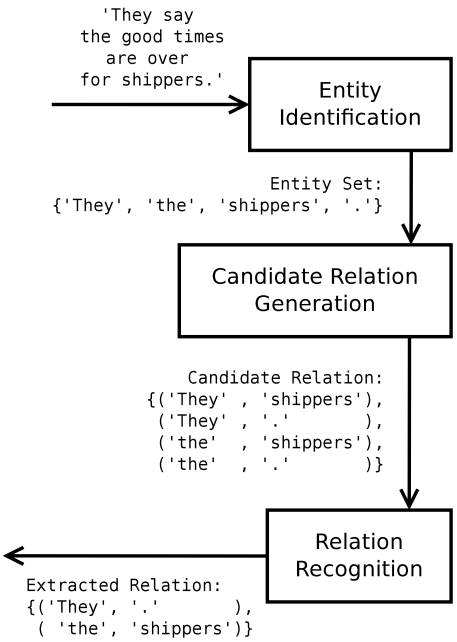

Fig. 2 Diagram of the *RelHunter* method

trate the application of *RelHunter* to clause identification. In this example, we use the sentence in Fig. 1.

Entity Identification is a local subtask, in which simple entities are detected without any concern about the structures they belong to. The outcome of this subtask is the *entity set*. For instance, for clause identification, we identify two types of entities: tokens that start a clause and tokens that end a clause.

The second subtask is performed by a simple procedure that generates the *candidate relation* over the entity set. This relation includes true and false *candidates*. This procedure considers domain specific knowledge to avoid the generation of all possible candidates. In the clause identification task, we generate a *binary* relation composed of all entity pairs (s, e) such that s is a start clause token, e is an end clause token, and s does not occur after e.

The Relation Recognition subtask is a binary classification problem in which *RelHunter* discriminates between true and false candidates. The output of this subtask is the *extracted relation* that contains the identified candidates.

In the following sections, we detail *RelHunter*'s design rationale. In Sect. 4, the task decomposition scheme is explained. In Sect. 5, we describe Interdependent Classification, an important issue explored by *RelHunter*.

4 Task decomposition

RelHunter decomposes the original task into three simpler subtasks that are sequentially solved. The first subtask, Entity Identification, is further decomposed into several token classification problems.

RelHunter tackles the nonlocal aspects of structured problems by using its task decomposition scheme. Entity Identification is just a local procedure, whereas the last two subtasks efficiently explore the global context of the target structures.

4.1 Entity identification

This is a local subtask in which simple entities are detected without any concern about the structures they belong to. An entity is given by a *type* and a set of consecutive tokens. Additionally, one of its tokens is defined as the *entity head*.

We decompose the Entity Identification subtask into several token classification tasks, one for each entity type. Thus, we use the original corpus to train several classifiers, also one per entity type. The outcome of this subtask is the entity set.

For instance, in the clause identification task, we consider two entity types: tokens that start a clause and tokens that end a clause. Since these entities comprise only one token, that one is the entity head. To identify these entities, we train two binary classifiers: one for the start token identification subtask and another for the end token identification subtask.

4.2 Candidate relation generation

In this intermediate subtask, we are given the entity set and hence use a domain-specific procedure to generate the candidate relation. Each instance within this relation represents a candidate structure. Thus, this relation includes true and false candidates.

For clause identification, we generate a binary relation with all entity pairs composed by a start clause token and an end clause token such that the start token does not occur after the end token.

4.3 Relation recognition

This is a classification subtask in which a binary classifier discriminates between true and false candidates within the candidate relation. To train this classifier, we build the *relation dataset* with an entry for each candidate. This dataset comprises two feature sets: *local* and *global*.

The local features are related to the candidate entities. For each entity, we just copy the features of its head token from the original corpus to the relation dataset.

A corpus is composed by *segments* whose definition is task dependent. For instance, in clause identification a segment is a sentence, whereas in quotation extraction a segment is a paragraph. The global features carry information of the candidate and its entities related to their whole segment.

For each candidate and each candidate entity, we split its segment into three *fragments*. Indeed, given a candidate, the

three fragments are the tokens before the candidate, the tokens within the candidate, and the tokens after the candidate. Similarly, for each entity, we split its segment into the tokens before the entity, the tokens within the entity, and the tokens after the entity.

The global features follow Carreras et al. [3]. These features inform about the occurrence of *relevant elements* within each considered fragment. The definition of a relevant element is application specific. In the clause identification task, we consider three relevant elements: verbal chunks, start clause tokens, and end clause tokens.

For each relevant element and fragment, we generate two global features in the relation dataset: a flag indicating the occurrence of the element within the fragment and a counter showing its frequency. Thus, the relation dataset has km local features and $6r(k+1)$ global features, where k is the relation cardinality, m is the number of features in the original corpus, and r is the number of relevant elements. For instance, the relation dataset for clause identification has 60 input features, since $k=2$, $m=3$, and $r=3$ for this task.

5 Interdependent classification

The input to the Relation Recognition subtask is the candidate relation, i.e., a set of candidates. The corresponding classifier must discriminate between true and false candidates. However, identifying one candidate as true implies that some other candidates must be false. This involves a special modeling issue: interdependent classification. The learning engine may explore these dependencies when building the classifier for this subtask.

Interdependent classification is usually assumed for neighboring examples. When the learning model adopts a Markovian property, the neighborhood is given by a *context window*. This is the case for Markovian fields such as hidden Markov models. Another ML model that performs interdependent classification is Entropy Guided Transformation Learning (ETL). *RelHunter* uses ETL as its basic learning engine.

Next, we describe ETL and show how *RelHunter* explores its interdependent classification modeling.

5.1 Entropy Guided Transformation Learning

ETL is a supervised ML algorithm for multiclass classification problems. ETL is error driven. The algorithm generates transformation rules to correct classification errors in the training corpus. ETL generalizes Transformation Based Learning (TBL) [1] by automatically generating rule templates. ETL employs an *entropy guided* template generation approach, which uses the information gain measure in order to select feature combinations that provide good rule template sets. In Fig. 3, we outline the ETL training algorithm.

1. Apply the baseline system to the training corpus.
2. Generate the rule templates by using an entropy-guided approach.
3. *Repeat*:
   (a) Generate, for each classification error in the current version of the training corpus, correcting rules by instantiating the templates.
   (b) Compute the rule scores. The rule score is defined as the difference between the total number of repaired errors and the total number of generated errors.
   (c) *Stop*, if there is no rule with a score greater than a given threshold.
   (d) Apply the best-scoring rule to the training corpus.
   (e) Append the best-scoring rule to the sequence of learned rules.
4. Return the sequence of learned rules.

Fig. 3 Entropy Guided Transformation Learning algorithm

For a detailed discussion of the ETL algorithm and the corresponding modeling approach, see [6, 16].

ETL has been successfully applied to part-of-speech tagging [7], phrase chunking, named entity recognition [6, 17], clause identification [10, 11], dependency parsing [18], and hedge detection [12], achieving results at least as good as the ones of TBL with handcrafted templates and close to state-of-the-art results. Several ETL-based multilingual systems are freely available on the Web through the F-EXT[1] service [9].

ETL uses an annotated dataset that is partitioned into *segments* such that each segment is a sequence of examples. Examples within the same segment are considered dependent. Conversely, examples within different segments are considered independent. Moreover, an example classification depends only on the features of the examples from its corresponding context window. The context window includes surrounding examples that may be considered when generating a rule to correct the classification of an example.

Hence, to apply ETL we need to provide three modeling ingredients: dataset segment definition, example ordering within a segment, and the context window size. Given that, classification dependencies are explored by the ETL classifier.

5.2 ETL modeling for relation recognition

We propose two different dataset segment definitions. The first one groups the candidates by their corresponding segments in the original dataset. For the second definition, we choose one of the example attributes and group in the

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

same segment all the candidates that share the same attribute value. A variation of this scheme is to group candidates by more than one attribute.

We use a very simple example ordering scheme. To order the candidates within a segment of the relation dataset, we apply the same order of their entities in the original corpus.

Given the segment definition and the example ordering, the ETL context window is defined just by its size. This number is task dependent and set by model tuning.

6 NLP tasks

In this section, we describe the *RelHunter* modeling for five natural language processing (NLP) tasks: clause identification, phrase chunking, hedge detection, quotation extraction, and dependency parsing. The multilingual tasks share the same modeling for the different corpora. For each task, we describe the task itself, the corresponding corpora, the entity identification, the candidate relation generation, the considered relevant elements, and the ETL parameters.

6.1 Phrase chunking

Phrase chunking [23] is a kind of shallow parsing and provides valuable information for important and more elaborated tasks, such as clause identification [24], dependency parsing [19], and semantic role labeling [14].

Phrase chunking consists in identifying nonoverlapping chunks of syntactically correlated words within a sentence. The three most important chunk types are (i) *nominal*—in which the head token is a noun; (ii) *verbal*—in which the head token is a verb; and (iii) *prepositional*—in which the head token is a preposition. Some corpora include more types, such as *adverbial* and *adjectival*, for instance.

6.1.1 Corpora

We apply *RelHunter* to two phrase chunking corpora: the English corpus provided in the CoNLL 2000 Shared Task [22] and the Portuguese corpus described in Fernandes et al. [11]. Both corpora include the same feature set: token word, part-of-speech (POS) tag, and phrase chunk. Nevertheless, the Portuguese corpus includes only the three most important chunk types. The English corpus includes some more chunk types. The *RelHunter* modeling does not use any language-specific knowledge. Both the English and Portuguese corpora are divided into three parts: training, development, and test.

6.1.2 Entity identification

For this task, we consider as entities the tokens that start or end a chunk. An entity also determines its chunk type. We train one ETL classifier to identify start tokens and another to identify end tokens. Both classifiers also identify the chunk types.

6.1.3 Candidate relation generation

Given the start and end tokens along with their chunk types, the binary candidate relation includes all the pairs of start and end tokens such that the start and end tokens lie within the same sentence, the start token does not occur after the end token, and both start and end tokens have the same chunk type.

In this task, we consider the following as relevant elements: verbal tokens, start tokens, and end tokens. Verbal tokens are identified by their POS tags. We group the candidates by their sentences and consider a context window with 7 candidates.

6.2 Clause identification

Clause identification consists in identifying all the clauses within a sentence. Clause identification is a kind of shallow parsing and has been chosen as the CoNLL 2001 Shared Task [24]. A preliminary version of the *RelHunter* method applied to clause identification is presented in [10, 11].

6.2.1 Corpora

We apply the *RelHunter* method to two clause corpora: the English corpus provided in the CoNLL 2001 Shared Task [24] and the Portuguese corpus introduced in Fernandes et al. [11]. The latter is based on the Bosque corpus [13]. These two corpora include very similar feature sets, and the *RelHunter* modeling does not use any language-specific knowledge. The two corpora include three input features: token word, POS tag, and phrase chunk. Both the English and Portuguese corpora are divided into three parts: training, development, and test.

6.2.2 Entity identification

We define two entity types for clause identification: tokens that start a clause and tokens that end a clause. In order to identify these entities, we train two simple binary ETL models, one for each entity type.

6.2.3 Candidate relation generation

The candidate relation is a binary relation that comprises all the start and end token pairs such that the two entities lie within the same sentence, and the start token does not occur after the end token. The relevant elements are verbal chunks, start tokens, and end tokens. We group the candidates by their sentence and consider a context window with 7 candidates.

6.3 Hedge detection

Hedges are linguistic devices that indicate uncertain or unreliable information within a text. The detection of hedge structures is important for many applications that extract facts from textual data. The CoNLL 2010 Shared Task [8] is dedicated to hedge detection.

A hedge structure consists of a *cue* and a *scope*. The hedge cue comprises one or more keywords that determine the uncertainty. The hedge scope is the uncertain statement which is hedged by the cue. The hedge scope always includes the corresponding cue.

6.3.1 Corpus

The corpus provided in the CoNLL 2010 Shared Task is based on the *BioScope corpus* [26]. This corpus includes scientific texts from the biological domain and is manually annotated with hedge cues and their scopes. We use state-of-the-art ETL systems [6, 10, 16] to include in this corpus the following features: POS tags, phrase chunks, and clause annotations. This corpus is divided into two parts: training and test. Besides that, the training part comprises scientific paper abstracts and full papers. To perform parameter tuning and model selection, we use the abstracts part as training corpus and the full papers part as development corpus. After the modeling phase, we evaluate our system on the test corpus.

6.3.2 Entity identification

We consider three types of hedge entities: cue chunk, start scope token, and end scope token. We train three ETL models to detect these entities.

The cue detection subtask is approached as a token classification problem by using the IOB tagging style. In this subtask, a token is classified as I when it is inside a hedge cue, as O when it is outside a hedge cue, and as B when it begins a hedge cue immediately after a distinct cue.

The start and end token scope subtasks are modeled as binary classification problems. We train an ETL token classifier to tackle each one of them.

6.3.3 Candidate relation generation

The hedge candidate relation contains the entity triples that comprise a hedge cue, a start scope token and an end scope token, such that the three entities lie within the same sentence, the start token does not occur after the end token and the cue chunk lie between the start and the end tokens. We choose verbal chunks and the three entities as relevant elements. We group the candidates by the cue and start scope entities. We consider a context window with 7 candidates.

6.4 Quotation extraction

Quotation extraction is an important task that has not been exploited much in the scientific literature. Indeed, to the best of our knowledge, there are no ML-based system and no publicly available annotated corpus for this task. Nevertheless, there are many systems available in the Web that provide, or internally use, this type of information. Almost all systems are based on manual pattern rules and gazetteers [5, 20, 25].

6.4.1 Corpus

The corpus used in this task comprises news feeds from the *New York Times*. It contains 69 news feeds, and 441 quotations have been manually annotated. We also include POS tags and Named Entity tags in this corpus. To evaluate the system performance, we perform a tenfold cross-validation procedure.

6.4.2 Entity identification

We consider two entity types: *person* and *quotation*. Person entities are directly detected from the named entity tags. Quotation entities are detected by a simple method that analyzes quotation marks.

6.4.3 Candidate relation generation

The candidate relation comprises all the pairs of person and quotation entities within the same paragraph. The following elements are considered as relevant: person entities, quotation entities, quotation verbs, personal pronouns, punctuation marks, and quotation marks. Quotation verbs are the ones that frequently indicate a quotation. We group the candidate relations by their quotation entity, since one quotation is stated by only one person among all possible candidates. Hence, the grouped candidates have a strong dependency: only one of them can be true.

6.5 Dependency parsing

The *dependency tree* of a sentence is a directed tree, such that each node corresponds to a token and an arc (h, d) indicates that the token d is syntactically dependent on the *head* token h. Dependency parsing [19] consists in deriving the dependency tree of a sentence by identifying the head of each token.

The dependency tree is regarded as an important information in computational linguistics. Hence, it has been chosen as the CoNLL Shared Task in 2006 and 2007 and part of the shared tasks in 2008 and 2009.

6.5.1 Corpus

We apply the *RelHunter* method to the Portuguese corpus provided in the CoNLL 2006 Shared Task [2]. This corpus is derived from the Bosque corpus [13]. The corpus also includes the following features: token word, word lemma, coarse-grained POS, fine-grained POS, and a list of set-valued syntactic and morphological features.

6.5.2 Entity identification

We define two entity types for this task: *dependent tokens* and *head tokens*. We have to predict the head token for each token in the corpus. Besides, for a given token, every token within the same sentence is a potential head. Thus, we do not need a classifier to identify entities, since all tokens are potential entities.

6.5.3 Candidate relation generation

For dependency parsing, we approach the Candidate Relation Generation by an ML method. We train two token classification models in order to perform this subtask. First, we train a binary ETL model to predict, for each token, at which side (left or right) its head token lies. We train another ETL token classification model to predict the POS tag of the head token.

To train these two ETL models, we use the input features from the original corpus and three derived features: the total number of verbs before the token, the total number of verbs after the token, and the lemma of the nearest verb before the token.

The binary candidate relation includes all the token pairs (h, d) such that h and d lie within the same sentence, and h lies on the predicted side of d and has the predicted POS tag. We group the relation candidates by the dependent token entity. Hence, all head candidates for a given token lie within the same segment in the relation dataset.

7 Experimental evaluation

We perform several experiments on seven multilingual corpora in order to evaluate the *RelHunter* performance and compare it to other systems. In Table 2, we show the major characteristics of the chosen corpora.

In Table 3, we present detailed evaluation results on these corpora. The presented results are divided into four column groups: (i) ETL token classification approaches; (ii) systems derived from a modified *RelHunter* method that does not perform interdependent classification; (iii) systems derived from the *RelHunter* method; and (iv) state-of-the-art systems. In each column group, we report precision, recall, and F-score. In the last column group, we also include the state-of-the-art system names.

The experimental evaluations are performed through each corpus test set, except for quotation extraction, where we use a tenfold cross-validation procedure. For the dependency parsing task, we report only the F-score, since this is a token classification problem and this measure is equal to precision, recall, and accuracy.

7.1 Token classification benchmarking

We compare *RelHunter* with ETL approaches based on token classification modeling. The performances of these to-

Table 2 Training corpora characteristics

Language	Task	Sizes		
		Segments	Tokens	Structures
Portuguese	Chunking	6,557	158,819	88,041
	Clause	6,557	158,819	14,767
	Dependency	9,071	206,678	206,678
English	Chunking	8,936	211,727	106,978
	Clause	8,936	211,727	24,841
	Hedge	12,818	338,299	2,892
	Quotation	69	61,512	441

Table 3 Detailed results: precision (P), recall (R), and F-score (F_1)

Language	Task	Token classification			*RelHunter* (no dependency)			*RelHunter*			State-of-the-art			
		P	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1	*P*	*R*	F_1	*System*
Portuguese	Chunking	86.44	85.79	86.11	90.41	83.98	87.08	90.41	83.98	87.08	90.41	83.98	87.08	*RelHunter*
	Clause	69.85	64.65	67.15	76.70	62.97	69.16	79.19	62.94	70.14	79.19	62.94	70.14	*RelHunter*
	Dependency	–	–	87.92	–	–	88.18	–	–	88.18	–	–	91.36	MST Parser [15]
English	Chunking	91.89	92.28	92.09	94.07	90.52	92.26	94.07	90.52	92.26	94.12	94.13	94.12	SVM [27]
	Clause	80.47	72.28	76.16	85.76	75.68	80.41	86.10	75.91	80.68	88.17	81.10	85.03	FRP [4]
	Hedge	52.24	53.16	52.69	57.84	50.73	54.05	57.84	50.73	54.05	59.62	55.18	57.32	Morante [8]
	Quotation	85.35	84.58	84.97	84.86	78.08	81.33	89.77	89.57	89.67	89.77	89.57	89.67	*RelHunter*

ken classification approaches are presented in the first column group of Table 3. The *RelHunter* systems outperform these approaches on the seven evaluated corpora. The average F_1 of the *RelHunter* systems is 2.14% greater than the average obtained by the token classification approaches; moreover, the average precision is 4.49% greater.

7.2 Interdependent classification

We apply a modified version of the *RelHunter* method that considers no dependency among the relation candidates, i.e., it uses a context window with only one candidate. These results are presented in the second column group of Table 3. The *dependency* version outperforms the *no dependency* version in three out of seven evaluated corpora. In the remaining corpora, the performance is exactly the same.

The average F_1 of the *dependency* version is 1.37% greater than the one achieved by the *no dependency* version. In this comparison, there is a large deviation for the Quotation Extraction task. For this task, the F_1 of the *dependency* version is 8.34% greater. We believe this is due to the strong dependency in the quotation relation. One specific quotation is stated by exactly one person and hence the relation instances that include the same quotation have a strong dependency: only one is true.

We believe that the null impact of the interdependent classification scheme on three tasks (phrase chunking, hedge detection, and dependency parsing) is due to the entity identification phase used for these tasks. The entity definition used in these tasks conveys strong information about the target structures. Hence, the entity detection phase becomes hard and the relation recognition phase becomes trivial. We believe that one may improve the *RelHunter* performance on these cases by modeling *weaker* entities and consequently transferring classification effort to the relation recognition phase.

7.3 State-of-the-art systems

We also compare the *RelHunter* performances with the performances achieved by state-of-the-art systems. The results show that *RelHunter* is valuable. It achieves state-of-the-art performance on three corpora and the performances on the other four corpora are close to the state-of-the-art performances.

8 Conclusion

For the past two decades, machine learning (ML) methods have been successfully applied to computational linguistic problems. Some of these problems, like part-of-speech tagging, are easily modeled as token classification problems. However, for many others, the task is to identify complex structures within a text. These structures frequently have two important aspects: they comprise many tokens, and they show nonlocal token dependencies. Despite that, many approaches cast such problems as token classification in order to ease the application of ML algorithms.

Here, we propose *RelHunter*, an ML-based method for the extraction of structured information from text. *RelHunter* consists in modeling the target structures as a relation over entities and thus decomposes the original task into several simpler subtasks. Task decomposition is a central idea in the *RelHunter* method and allows it to efficiently tackle the nonlocal aspects of complex problems.

RelHunter is based on the Entropy Guided Transformation Learning (ETL) algorithm. Like hidden Markov models, ETL performs interdependent classification, a powerful feature exploited by *RelHunter* to consider dependencies among the target structures. Experimental results indicate that this technique is indeed effective.

RelHunter may be easily applied to many complex computational linguistic problems. We show its versatility by applying it to five multilingual tasks: phrase chunking, clause identification, hedge detection, quotation extraction, and dependency parsing.

We evaluate *RelHunter*-based systems on seven corpora and compare their performances with the corresponding token classification and state-of-the-art methods. The proposed systems are effective and outperform the token classification systems on all evaluated corpora. Moreover, *RelHunter* systems achieve state-of-the-art results for three corpora and, for the other four corpora, achieve good results.

There are several other computational linguistic problems that may be easily approached by using *RelHunter*. Semantic role labeling [14] is a very complex and important example that would contribute to a better understanding and use of *RelHunter*.

RelHunter explores the dependency among language structures by using a powerful feature of the ETL algorithm. Nevertheless, this feature is restricted to linearly organized examples, since ETL has been initially proposed for token classification problems. Language structures involve dependency topologies that are frequently much more complex than that. The ETL algorithm may be extended to consider more complex dependency topologies. We conjecture that it is possible to consider quite general topologies. This would contribute to the construction of better solutions to many computational linguistic tasks.

Acknowledgements The authors thank Carlos E.M. Crestana for providing evaluation results, algorithms, and other resources that are very helpful when solving the Dependency Parsing task. The first author also thanks the Instituto Federal de Educação, Ciência e Tecnologia de Goiás for their financial support.

References

1. Brill E (1995) Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. Comput Linguist 21(4):543–565
2. Buchholz S, Marsi E (2006) CoNLL-X shared task on multilingual dependency parsing. In: Proceedings of the tenth conference on computational natural language learning, New York, USA, pp 149–164
3. Carreras X, Màrquez L, Punyakanok V, Roth D (2002) Learning and inference for clause identification. In: Proceedings of the thirteenth European conference on machine learning, pp 35–47
4. Carreras X, Màrquez L, Castro J (2005) Filtering-ranking perceptron learning for partial parsing. Mach Learn 60(13):41–71
5. de La Clergerie É, Sagot B, Stern R, Denis P, Recourcé G, Mignot V (2009) Extracting and visualizing quotations from news wires. In: Proceedings of the 4th language and technology conference, Poznań, Poland, November
6. dos Santos CN, Milidiú RL (2009) Entropy guided transformation learning. In: Foundations of computational intelligence, volume 1: Learning and approximation. Studies in Computational Intelligence, vol 201. Springer, Berlin, pp 159–184
7. dos Santos CN, Milidiú RL, Renteria RP (2008) Portuguese part-of-speech tagging using entropy guided transformation learning. In: Proceedings of the international conference on computational processing of Portuguese language (PROPOR), Aveiro, Portugal
8. Farkas R, Vincze V, Mora G, Csirik J, Szarvas G (2010) The CoNLL 2010 shared task: learning to detect hedges and their scope in natural language text. In: Proceedings of the fourteenth conference on computational natural language learning shared task (CoNLL), Uppsala, Sweden
9. Fernandes ER, dos Santos CN, Milidiú RL (2009) Portuguese language processing service. In: Proceedings of the web in Ibero-America alternate track of the 18th World Wide Web conference (WWW), Madrid
10. Fernandes ER, Pires BA, dos Santos CN, Milidiú RL (2009) Clause identification using entropy guided transformation learning. In: Proceedings of the 7th Brazilian symposium in information and human language technology (STIL), São Carlos, Brazil
11. Fernandes ER, Pires BA, dos Santos CN, Milidiú RL (2010) A machine learning approach to Portuguese clause identification. In: Proceedings of the 9th international conference on computational processing of the Portuguese language (PROPOR), Porto, Alegre, Brazil. Lecture notes in artificial intelligence, vol 6001. Springer, Berlin, pp 55–64
12. Fernandes E, Crestana C, Milidiú R (2010) Hedge detection using the RelHunter approach. In: Proceedings of the 14th conference on computational natural language learning, July 2010, Uppsala, Sweden. Association for Computational Linguistics, Stroudsburg, pp 64–69. http://www.aclweb.org/anthology/W10-3009
13. Freitas MC, Rocha P, Bick E (2008) Floresta sintá(c)tica: bigger, thicker and easier. In: Teixeira A, Lúcia Strube de Lima V, Caldas de Oliveira L, Quaresma P (eds) Computational processing of the Portuguese language. Lecture notes in computer science, vol 5190. Springer, Berlin, pp 216–219
14. Màrquez L, Carreras X, Litkowski KC, Stevenson S (2008) Semantic role labeling: an introduction to the special issue. Comput Linguist 34(2):145–159
15. McDonald R, Lerman K, Pereira F (2006) Multilingual dependency analysis with a two-stage discriminative parser. In: Proceedings of the tenth conference on computational natural language learning, New York, USA. Association for Computational Linguistics, Stroudsburg, pp 216–220
16. Milidiú RL, dos Santos CN, Duarte JC (2008) Phrase chunking using entropy guided transformation learning. In: Proceedings of ACL–HLT, Columbus, OH, USA. Association for Computational Linguistics, Stroudsburg, pp 647–655
17. Milidiú RL, dos Santos CN, Duarte JC (2008) Portuguese corpus-based learning using ETL. J Braz Comput Soc 14(4). doi:10.1590/S0104-65002008000400003
18. Milidiú RL, dos Santos CN, Crestana CEM (2009) A token classification approach to dependency parsing. In: Proceedings of the 7th Brazilian symposium in information and human language technology (STIL), São Carlos, Brazil
19. Nivre J, Hall J, Kübler S, McDonald R, Nilsson J, Riedel S, Yuret D (2007) The CoNLL 2007 shared task on dependency parsing. In: Proceedings of the CoNLL shared task, Prague, Czech Republic, pp 915–932
20. Pouliquen B, Steinberger R, Best C (2007) Automatic detection of quotations in multilingual news. In: Proceedings of recent advances in natural language processing, Borovets, Bulgaria, September
21. Punyakanok V, Roth D (2001) The use of classifiers in sequential inference. In: Proceedings of the conference on advances in neural information processing systems (NIPS). MIT Press, Cambridge, pp 995–1001
22. Sang EFTK (2000) Text chunking by system combination. In: Proceedings of conference on computational natural language learning, Lisbon, Portugal
23. Sang EFTK, Buchholz S (2000) Introduction to the CoNLL-2000 shared task: chunking. In: Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal
24. Sang EFTK, Déjean H (2001) Introduction to the CoNLL-2001 shared task: clause identification. In: Proceedings of fifth conference on computational natural language learning, Toulouse, France
25. Sarmento L, Nunes S (2009) Automatic extraction of quotes and topics from news feeds. In: Proceedings of the 4th doctoral symposium on informatics engineering, Porto, Portugal, February
26. Vincze V, Szarvas G, Richárd F, Mora G, Csirik J (2008) The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. BMC Bioinf 9(Suppl 11):S9
27. Wu YC, Chang CH, Lee YS (2006) A general and multi-lingual phrase chunking model based on masking method. In: Proceedings of the 7th international conference on intelligent text processing and computational linguistics, pp 144–155