

Hedge Detection using the *RelHunter* Approach*

Eraldo R. Fernandes[†] and Carlos E. M. Crestana[‡] and Ruy L. Milidiú[§]

Departamento de Informática, PUC-Rio
Rio de Janeiro, Brazil

efernandes, ccrestana, milidiu@inf.puc-rio.br

Abstract

RelHunter is a Machine Learning based method for the extraction of structured information from text. Here, we apply *RelHunter* to the Hedge Detection task, proposed as the CoNLL 2010 Shared Task¹. *RelHunter*'s key design idea is to model the target structures as a relation over entities. The method decomposes the original task into three subtasks: (i) Entity Identification; (ii) Candidate Relation Generation; and (iii) Relation Recognition. In the Hedge Detection task, we define three types of entities: *cue chunk*, *start scope token* and *end scope token*. Hence, the Entity Identification subtask is further decomposed into three token classification subtasks, one for each entity type. In the Candidate Relation Generation subtask, we apply a simple procedure to generate a *ternary* candidate relation. Each instance in this relation represents a hedge candidate composed by a cue chunk, a start scope token and an end scope token. For the Relation Recognition subtask, we use a binary classifier to discriminate between true and false candidates. The four classifiers are trained with the *Entropy Guided Transformation Learning* algorithm. When compared to the other hedge detection systems of the CoNLL shared task, our scheme shows a competitive performance. The *F*-score of our system is 54.05 on the evaluation corpus.

* This work is partially funded by CNPq and FAPERJ grants 557.128/2009-9 and E-26/170028/2008.

[†] Holds a CNPq doctoral fellowship and has financial support from IFG, Brazil.

[‡] Holds a CAPES doctoral fellowship.

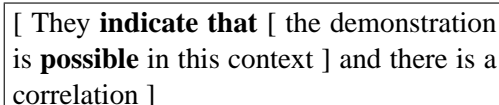
[§] Holds a CNPq research fellowship.

¹ *Closed Task 2*: detection of hedge cues and their scopes.

1 Introduction

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 (Farkas et al., 2010) is dedicated to hedge detection.

A hedge structure consists of a *cue* and a *scope*. In Figure 1, we present a sentence with two hedge instances. The hedge cues are highlighted and their scopes are delimited by brackets. The hedge cue comprises one or more keywords that indicate uncertainty. The hedge scope is the uncertain statement which is hedged by the cue. The scope always includes the corresponding cue.



[They **indicate that** [the demonstration is **possible** in this context] and there is a correlation]

Figure 1: Sentence with two hedge instances.

Over the last two decades, several Computational Linguistic problems have been successfully modeled as local token classification tasks (Brill, 1995; Milidiú et al., 2009). Nevertheless, the harder problems consist in identifying complex structures within a text. These structures comprise many tokens and show non local token dependencies.

Phrase chunking (Sang and Buchholz, 2000) is a task that involves structure recognition. Panyakanok and Roth decompose this task into four subtasks, that are sequentially solved (Panyakanok and Roth, 2001). They use Hidden Markov Models for the first three subtasks. They find out that task decomposition improves the overall token classification modeling.

Clause identification (Sang and Déjean, 2001) is another task that requires structure recognition. As clauses may embed other clauses, these struc-

tures involve stronger dependencies than phrase chunks. Carreras *et al.* propose an approach that extends Punyakanok and Roth’s previous work (Carreras *et al.*, 2002). Their system comprises complex methods for training and extraction, in order to exploit the specific dependency aspects of clause structures.

Phrase Recognition is a general type of task that includes both phrase chunking and clause identification. Carreras *et al.* propose the Filtering-Ranking Perceptron (FRP) system for this general task (Carreras *et al.*, 2005). The FRP task modeling is strongly related to previous proposals (Punyakanok and Roth, 2001; Carreras *et al.*, 2002). However, it simultaneously learns to solve three subtasks. FRP is very effective, although computationally expensive at both training and prediction time. Currently, FRP provides the best performing clause identification system.

In (Morante and Daelemans, 2009), the hedge detection task is solved as two consecutive classification tasks. The first one consists of classifying the tokens of a sentence as hedge cues using the IOB tagging style. The second task consists of classifying tokens of a sentence as being the start of a hedge scope, the end of one, or neither. The result of those two tasks is combined using a set of six rules to solve the hedge detection task.

Here, we describe *RelHunter*, a new method for the extraction of structured information from text. Additionally, we apply it to the Hedge Detection task. *RelHunter* extends the modeling strategy used both in (Carreras *et al.*, 2005) and (Punyakanok and Roth, 2001). Other applications of this method are presented in (Fernandes *et al.*, 2009b; Fernandes *et al.*, 2010).

The remainder of this text is organized as follows. In Section 2, we present an overview of the *RelHunter* method. The modeling approach for the Hedge Detection task is presented in Sections 3 and 4. The experimental findings are depicted and discussed in Section 5. Finally, in Section 6, we present our final remarks.

2 RelHunter Overview

The central idea of *RelHunter* is to model the target structures as a relation over entities. To learn how to extract this relation from text, *RelHunter* uses two additional schemes: task decomposition and interdependent classification.

We decompose the original task into three sub-

tasks: (i) *Entity Identification*; (ii) *Candidate Relation Generation*; and (iii) *Relation Recognition*. In Figure 2, we illustrate the application of *RelHunter* to hedge detection. We use the sentence introduced by Figure 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 hedge detection, we identify three types of entities: hedge cues, tokens that start a scope and tokens that end a scope.

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 hedge detection task, we define the candidate relation as the set of entity triples that comprise a hedge cue, a start scope token and an end scope token, such that the start token does not occur after the end token and the hedge cue occurs between the start and the end tokens.

The Relation Recognition subtask is a binary classification problem. In this subtask, we discriminate between true and false candidates. The *output relation* produced in this subtask contains the identified hedge instances.

3 Hedge Detection using RelHunter

In this section, we detail the *RelHunter* method and describe its application to hedge detection.

3.1 Entity Identification

We consider three specific entity types: cue chunk, start scope token, and end scope token. We divide entity identification into three token classification tasks, one for each entity type. Thus, we use the original corpus to train three classifiers.

The *cue chunk* subtask is approached as a token classification problem by using the IOB tagging style. The token tag is defined as follows: I, when it is inside a hedge cue; O, when it is outside a hedge cue; and B, when it begins a hedge cue immediately after a distinct cue. As the baseline classifier, we use the Cue Dictionary proposed in (Morante and Daelemans, 2009), classifying each occurrence of those words as a cue.

The *start scope* and *end scope* subtasks are modeled as binary token classification problems.

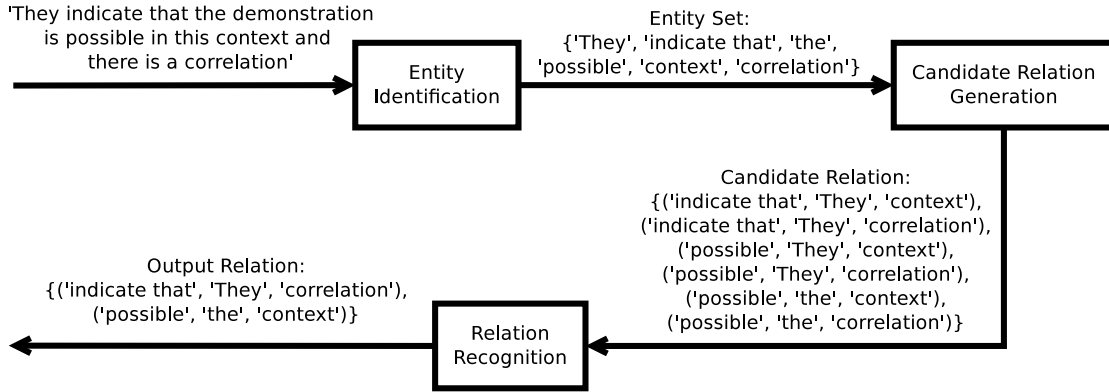


Figure 2: Diagram of the *RelHunter* method.

As the baseline classifier for the start scope subtask, we assign the first token of each hedge cue as the start of a scope.

We have two baseline classifiers for the end scope subtask: *END* and *END-X*. The *END* system classifies as an end token the second to the last token of each sentence that contains a cue. Due to the frequent occurrence of parenthesized clauses at the end of sentences in full articles, the *END-X* system extends the *END* system with an additional operation. It reassigns an end scope tag, from a close parentheses token, to the token before its corresponding open parentheses.

3.2 Candidate Relation Generation

We define as the candidate hedge relation the set of entity triples that comprise a hedge cue, a start scope token and an end scope token, such that the start token does not occur after the end token and the hedge cue occurs between the start and the end tokens.

3.3 Relation Recognition

We train a binary classifier to discriminate between positive and negative candidates within the candidate relation. This classifier is trained on the *relation dataset*, which is built by a general procedure. This dataset contains an entry for each candidate. For each candidate, we generate two feature sets: *local features* and *global features*.

The local features include local information about each candidate entity, namely: cue chunk, start scope token and end scope token. These features are retrieved from the original corpus. For the start and end tokens, we use all their features in the original corpus. For the cue chunk, we use the features of the rightmost token within the chunk.

The global features follow Carreras *et al.* (Carreras et al., 2002). These features are generated by considering the whole sentence where the candidate lies in. They inform about the occurrence of *relevant elements* within sentence *fragments*. We consider as relevant elements the three entity types and verbal chunks.

For each candidate entity, we consider three fragments. The first one contains all the tokens before the entity. The second, all the entity tokens, and the third all the tokens after the entity. Similarly, for the whole candidate, we have three more fragments: one containing all the tokens before the candidate, another containing all the candidate tokens, and the third one containing all the tokens after the candidate. Thus, there are 12 fragments for each candidate, three for each entity plus three for the whole candidate.

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.

The relation dataset has km local features and $6r(k+1)$ global features, where k is the relation cardinality (number of entities), m is the number of features in the original corpus, and r is the number of relevant elements.

Our current *RelHunter* implementation uses the Entropy Guided Transformation Learning (ETL) as its learning engine (Milidiú et al., 2008; dos Santos and Milidiú, 2009). For instance, we train four ETL based classifiers: one for each Entity Identification subtask and one for the Relation Recognition subtask. In the next section, we describe an important issue explored by the ETL algorithm.

4 Interdependent Classification

The input to the Relation Recognition subtask is the candidate relation, i.e., a set of hedge candidates. The corresponding classifier must discriminate positive from negative candidates. However, identifying one candidate as positive implies that some other candidates must be negatives. 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, then the neighborhood is given by a context window. This is the case for Markovian Fields such as Hidden Markov Models. Another model that also explores interdependent examples is ETL.

ETL is a very attractive modeling tool and has been applied to several classification tasks (Milidiú et al., 2008; dos Santos and Milidiú, 2009; Fernandes et al., 2009a; Fernandes et al., 2010). ETL uses an annotated corpus, where the corresponding class is attached to each example. The corpus is partitioned into segments. 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*. Hence, to apply ETL we need to provide three modeling ingredients: segment definition, example ordering within a segment and the context window size. Given that, classification dependencies are explored by the ETL classifier. Hence, *RelHunter* uses ETL as its learning engine.

We include in the same segment the hedge candidates that have the same cue and start scope tokens. Within a segment, we order the candidates by the order of the end token in the original corpus. We use a context window of 7 candidates, i.e., three candidates before the current, the current candidate and three candidates after the current.

5 Experimental Results

We use the corpus provided in the CoNLL 2010 Shared Task to train and evaluate our hedge detection system. We add the following annotation to the corpus: word stems, part-of-speech tags, phrase chunks, and clause annotations. Word

stems have been generated by the Porter stemmer (Porter, 1980). The additional annotation has been generated by ETL based systems (dos Santos and Milidiú, 2009; Fernandes et al., 2009b; Milidiú et al., 2008).

The CoNLL corpus is based on the *BioScope* corpus (Vincze et al., 2008). Since it contains documents of two different kinds – paper abstracts and full papers – we split it into two subcorpora. The first subcorpus is called *ABST* and contains all the paper abstracts. The second is called *FULL* and contains all the full papers.

We have two experimental setups: *Development* and *Evaluation*. In the Development Setup, we use *ABST* as the training corpus and *FULL* as the development corpus. This is a conservative decision since the CoNLL Evaluation Corpus is comprised only of full articles. In the Evaluation Setup, we use the union of *ABST* and *FULL* as the training corpus and report the performance over the CoNLL Evaluation Corpus.

5.1 Development

Here, we report the development setup experimental findings. In Table 1, we show the performance of the three baseline classifiers. The start and end classifiers are evaluated with golden standard cue chunks. All results are obtained with the *END-X* baseline system, except when explicitly stated.

Table 1: Development performance of the three Baseline Classifiers.

<i>Task</i>	<i>Precision</i>	<i>Recall</i>	<i>F-score</i>
<i>Cue</i>	51.96	51.65	51.80
<i>Start scope</i>	72.01	72.22	72.11
<i>End scope</i>	65.90	58.97	62.24

In Table 2, we report the performance of the three entity identification ETL classifiers. Again, the start and end classifiers are evaluated with golden standard cue chunks. These results indicate that the end scope subtask is the hardest one. Indeed, our ETL classifier is not able to improve the baseline classifier performance. The last table line shows the performance of the *RelHunter* method on the target task – hedge detection.

5.2 Evaluation

Here, we report the evaluation setup findings. In Table 3, we show the performance of the three

Table 2: Development performance of the three entity identification ETL classifiers and the *RelHunter* method to hedge detection.

<i>Task</i>	<i>Precision</i>	<i>Recall</i>	<i>F-score</i>
<i>Cue</i>	81.23	73.20	77.01
<i>Start scope</i>	91.81	72.37	80.94
<i>End scope</i>	65.90	58.97	62.24
<i>Hedge</i>	53.49	34.43	41.89

baseline classifiers. The start and end classifiers are evaluated with golden standard cue chunks.

Table 3: Evaluation performance of the three Baseline Classifiers.

<i>Task</i>	<i>Precision</i>	<i>Recall</i>	<i>F-score</i>
<i>Cue</i>	45.12	60.02	51.52
<i>Start scope</i>	75.51	75.73	75.62
<i>End scope</i>	81.01	72.56	76.55

In Table 4, we report the performance of the three entity identification ETL classifiers. Again, the start and end classifiers are evaluated with golden standard cue chunks. The last table line shows the performance of the *RelHunter* method on the target task – hedge detection.

Table 4: Evaluation performance of the three entity identification ETL classifiers and the *RelHunter* method to hedge detection.

<i>Task</i>	<i>Precision</i>	<i>Recall</i>	<i>F-score</i>
<i>Cue</i>	78.73	77.05	77.88
<i>Start scope</i>	89.21	77.86	83.15
<i>End scope</i>	81.01	72.56	76.55
<i>Hedge</i>	57.84	50.73	54.05

In Table 5, we report the Hedge Detection performances when using *END* and *END-X*, as the baseline classifier for the end scope subtask. The use of *END-X* improves the overall system *F*-score by more than ten twelve.

In Table 6, we report the Final Results of the CoNLL 2010 Shared Task – Closed Task 2. For the sake of comparison, we also include the performance of the *RelHunter* system with *END-X*, that has been developed and tested after the com-

Table 5: Evaluation performance of the *RelHunter* system when using *END* and *END-X*.

<i>End scope</i>	<i>Precision</i>	<i>Recall</i>	<i>F-score</i>
<i>END</i>	45.96	38.04	41.63
<i>END-X</i>	57.84	50.73	54.05

petition end. The version with the *END* baseline holds rank 7 at the competition.

Table 6: Evaluation performance of the CoNLL 2010 systems and the *RelHunter* method with the *END-X* end scope classifier.

<i>Official Rank</i>	<i>System</i>	<i>P</i>	<i>R</i>	<i>F</i>
1	Morante	59.62	55.18	57.32
2	Rei	56.74	54.60	55.65
3	Velldal	56.71	54.02	55.33
-	<i>RelHunter</i>	57.84	50.73	54.05
4	Xinxin	57.42	47.92	52.24
5	Zhou	45.32	43.56	44.42
6	Shaodian	45.94	42.69	44.25
7	Fernandes	45.96	38.04	41.63
8	Vlachos	41.18	35.91	38.37
9	qzhao	34.78	41.05	37.66
10	BuzhouTang	34.49	31.85	33.12
11	Qiu	21.87	17.23	19.27
12	Täckström	02.27	02.03	02.15

6 Conclusion

We propose *RelHunter*, a new machine learning based method for the extraction of structured information from text. *RelHunter* consists in modeling the target structures as a relation over entities. To learn how to extract this relation from text, *RelHunter* uses two main schemes: task decomposition and interdependent classification.

RelHunter decomposes the identification of entities into several but simple token classification subtasks. Additionally, the method generates a candidate relation over the identified entities and discriminates between true and false candidates within this relation.

RelHunter uses the Entropy Guided Transformation Learning algorithm as its learning engine. As Hidden Markov Models, ETL is able to consider interdependent examples. *RelHunter* ex-

ploits this powerful feature in order to tackle dependencies among the hedge candidates.

RelHunter is easily applied to many complex Computational Linguistic problems. We show its effectiveness by applying it to hedge detection. Other successful applications of this method are presented in (Fernandes et al., 2009b; Fernandes et al., 2010).

RelHunter explores the dependency among linguistic structures by using a powerful feature of the ETL algorithm. Nevertheless, this feature is restricted to sequentially organized examples, since ETL has been initially proposed for token classification problems. Linguistic structures involve topologies that are frequently more complex than that. The ETL algorithm may be extended to consider more complex 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.

Acknowledgments

The authors thank Evelin Amorim and Eduardo Motta for coding dataset normalization procedures that are very handy for Hedge Detection.

References

- Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.
- Xavier Carreras, Lluís Màrquez, Vasin Punyakanok, and Dan Roth. 2002. Learning and inference for clause identification. In *Proceedings of the Thirteenth European Conference on Machine Learning*, pages 35–47.
- Xavier Carreras, Lluís Màrquez, and Jorge Castro. 2005. Filtering-ranking perceptron learning for partial parsing. *Machine Learning*, 60(1–3):41–71.
- Cícero N. dos Santos and Ruy L. Milidiú, 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.
- Richard Farkas, Veronika Vincze, Gyorgy Mora, Janos Csirik, and Gyorgy Szarvas. 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.
- Eraldo R. Fernandes, Cícero N. dos Santos, and Ruy L. Milidiú. 2009a. Portuguese language processing service. In *Proceedings of the Web in Ibero-America Alternate Track of the 18th World Wide Web Conference (WWW'2009)*, Madrid.
- Eraldo R. Fernandes, Bernardo A. Pires, Cícero N. dos Santos, and Ruy L. Milidiú. 2009b. Clause identification using entropy guided transformation learning. In *Proceedings of the 7th Brazilian Symposium in Information and Human Language Technology (STIL)*, São Carlos, Brazil.
- Eraldo R. Fernandes, Bernardo A. Pires, Cícero N. dos Santos, and Ruy L. Milidiú. 2010. A machine learning approach to portuguese clause identification. In *Proceedings of the Ninth International Conference on Computational Processing of the Portuguese Language (PROPOR)*, volume 6001 of *Lecture Notes in Artificial Intelligence*, pages 55–64, Porto Alegre, Brazil. Springer.
- Ruy L. Milidiú, Cícero N. dos Santos, and Julio C. Duarte. 2008. Phrase chunking using entropy guided transformation learning. In *Proceedings of ACL-08: HLT*, pages 647–655, Columbus, USA. Association for Computational Linguistics.
- Ruy L. Milidiú, Cícero N. dos Santos, and Carlos E. M. Crestana. 2009. A token classification approach to dependency parsing. In *Proceedings of the 7th Brazilian Symposium in Information and Human Language Technology (STIL'2009)*, São Carlos, Brazil.
- Roser Morante and Walter Daelemans. 2009. Learning the scope of hedge cues in biomedical texts. In *Proceedings of the BioNLP 2009 Workshop*, pages 28–36, Boulder, USA, June. Association for Computational Linguistics.
- M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.
- Vasin Punyakanok and Dan Roth. 2001. The use of classifiers in sequential inference. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 995–1001. MIT Press.
- Erik F. T. K. Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, Lisbon, Portugal.
- Erik F. T. K. Sang and Hervé Déjean. 2001. Introduction to the CoNLL-2001 shared task: Clause identification. In *Proceedings of Fifth Conference on Computational Natural Language Learning*, Toulouse, France.
- Veronika Vincze, György Szarvas, Richárd Farkas, György Mora, and János Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC Bioinformatics*, 9 (Suppl 11):S9.