

# Deriving a Statistical Syntactic Parsing from a Treebank

Mihaela Colhon<sup>\*</sup>

University of Craiova, Department of Computer Science

A.I. Cuza street, no. 13, 200585  
Craiova, Romania  
mcolhon@inf.ucv.ro

Radu Simionescu<sup>†</sup>

A. I. Cuza University of Iași, Faculty of Computer Science

Carol I street, no. 11, 700506  
Iași, Romania  
radu.simionescu@info.uaic.ro

## ABSTRACT

The study presented in this article is dedicated to a syntactic parser for Romanian. The central goal of the presented technique is to learn a model which is able to discriminate between probability for a word to be head of another word in a dependency structure corresponding to a sentence in the considered language. The model described in this paper was trained on a dependency treebank linguistic resource and is intended to be used in order to develop a dependency syntactic parser.

## Categories and Subject Descriptors

I.2 [Artificial Intelligence]: I.2.7 Natural Language Processing—*Language parsing and understanding*; I.2 [Artificial Intelligence]: I.2.6 Learning—*Parameter learning*

## Keywords

Syntactic parsing, Dependency relations, Maximum Entropy Model, Romanian language

## 1. INTRODUCTION

With the rise of the Semantic Web and the corresponding meta-data description language RDF, it became clear that Natural Language Processing (NLP) could be of enormous importance for the Semantic Web [10]. Indeed, NLP is vital to the success of the Semantic Web because it is

<sup>\*</sup>Mihaela Colhon is currently Assistant Professor and post-doctoral researcher at University of Craiova. Her personal web page is <http://inf.ucv.ro/~ghindeanu/en/index.html>.

<sup>†</sup>Radu Simionescu is a PhD student. He is currently engaged in the Metanet4U project. He is a former member of the Clarin project (the preparatory phase). He is the developer of various NLP tools from which the following are worth mentioning: a Romanian part of speech tagger (<http://nlptools.info.uaic.ro/WebPosTagger/>); the GGS matching tool (<http://sourceforge.net/projects/ggs/>) which also includes a grammar for Romanian NP chunking.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WIMS'12 June 13-15, 2012 Craiova, Romania

Copyright 2012 ACM 978-1-4503-0915-8/12/06 ...\$10.00.

the method of communication between humans and software agents [13]. For this reason, several NLP applications such as parsing, knowledge representation, Information Extraction and semantic analysis are used in many Semantic Web technologies.

During the last years, the efforts in the NLP community have been devoted to the semantic and syntactic studies of natural language constructs. The model presented in this article is dedicated to a syntactic parsing study with dependency structures.

As noted by competent linguists, Romanian language is morphologically rich which makes it a relatively free word order language. The term Morphologically Rich Languages (MRLs) refers to languages in which substantial grammatical information, i.e., information concerning the arrangement of words into syntactic units or cues to syntactic relations, are expressed at word level. Because of its rich morphology, the morphological markers themselves could serve as strong cues for identifying the syntactic relations between the words in a sentence. Still, in languages with free word-order, such as Romanian, constituency-based representations are overly constrained, this fact causing word-order choice to influence the complexity of the syntactic analysis.

Various machine learning techniques have been developed and successfully used in the syntactic NLP applications. As a result, many variants of the basic approaches using different features and different classifiers have been introduced, but the Maximum Entropy framework [11] pales all others by integrating facts without feature independence hypothesis [14].

The syntactic parsing is the task of deriving the syntactic structure corresponding to a natural language sentence. Two main approaches are used to describe a syntactic structure: *constituent parsing* and *dependency parsing*.

In this paper an unlabeled dependency parsing statistical model for a less-studied language, the Romanian language, is described. In this work we explore the dependency structures manually encoded in a Romanian Dependency Treebank developed at A.I.Cuza University of Iași by the Natural Language Processing Group of Faculty of Computer Science. The model can be classified as a Maximum Entropy model and simultaneously uses many context “features” to predict which words are heads or governors in a dependency structure.

## 1.1 Dependency relations

At the heart of every sentence structure are the relations among words, no matter if by these relations we mean

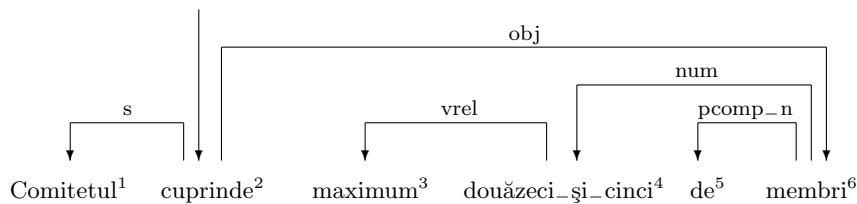


Figure 1: Labeled Dependency Graph

the possible grammatical functions or the links which bind words into larger syntactical units such as, noun phrases, verb phrases, etc. Usually, these relations are formalized by means of dependency grammar rules where each word is considered to be dependent on another word which links it to the rest of the sentence [12].

Dependency parsing is the task of deriving a syntactic dependency structure. In this structure each token of a sentence has a link (dependency) to a head token, except the root token that has no head.

There has been a growing interest in dependency parsing in recent years. For example, in [5] the authors found that the dependency structures of a pair of translated sentences have a greater degree of cohesion than phrase structures. Following this idea, in [2] the authors exploited such cohesion between the dependency structures to improve the quality of word alignment of parallel sentences.

A variety of dependency relations may exist among the words of a sentence if no restrictions are specified. No matter what language it means to define, any dependency grammar should take into account a number of linguistically motivated general principles. These principles are:

- any word should depend exactly on one word (the *head*) with the exception of the main predicate of the sentence and all the punctuation characters which do not depend on any other words
- several words may depend on the same head
- if the relations between the dependent and the head are represented by an arc oriented from head to dependent, then these arches should not intersect, and the oriented graph which is thus formed should not contain cycles

Given a sentence  $s = (w_1, w_2, \dots, w_n)$ , the dependency parsing results in head-modifier relations between pairs of words, together with the labels of the relationships. The labels describe the type of the relation, e.g. *subject*, *object* or *determiner*. The dependency representation of a sentence, consisting of its lexical elements linked by binary asymmetric relations forms a labeled directed graph (see Figure 1) or an unlabeled directed graph (for the case when the dependency arcs are unlabeled).

The aim of dependency parsing is to compute a tree structure of a sentence where nodes are words, and edges represent the relations among words. An important characteristic of the dependency graphs is their *projectivity* property.

The projectivity property of a dependency grammar is more appropriate for some languages than others. For example, in English, projective trees are sufficient to analyze most sentence types. In reality, there are correct unprojective examples for almost any language. Especially, in languages with more flexible word order than English, such as German, Dutch or Romanian, non-projective dependencies are more frequent.

Nevertheless, from the syntactic parsers point of view, parsing a sentence using the restriction of projectivity gives a small advantage over unprojective parsing. Still, the projectivity restriction, if applied consistently, induces some errors (for structures which are non-projective).

In the process of projective parsing, the number of possible heads of a word is restricted because of its dependencies that have been settled so far in the dependency tree. In unprojective parsing, this does not happen, which usually determines greater ambiguity in the parsing process. For these reasons, projective parsing is usually preferred, but unprojective parsing is considered closer to the human parser.

## 1.2 Treebank Data

In the last years, large corpora that allow researches in NLP to automatically extract information about natural languages were developed. At this moment, such resources are considered critical tools for any NLP system. The syntactically annotated treebanks facilitated the way for an alternative parsing natural languages paradigm: the data-driven (or statistical) parsing. Statistical approaches require less effort for building parsers, although their performance is directly related to the quality and size of the used treebank.

A variety of treebanks are already available for many languages (namely English<sup>1</sup>, German<sup>2</sup>, Czech<sup>3</sup> and Swedish<sup>4</sup> languages) but only few for Romanian language.

The parser presented in this article was train and tested on a Romanian Dependency Treebank developed at A.I.Cuza University of Iași by the Natural Language Processing Group of Faculty of Computer Science. The corpus is XML encoded obeying a simplified form of the XCES standard [6]. In the

<sup>1</sup>Penn Treebank: <http://www.cis.upenn.edu/~treebank/>

<sup>2</sup>NEGRA Treebank: <http://www.coli.uni-sb.de/sfb378/negra-corpus>

<sup>3</sup>Prague Dependency Treebank: <http://ufal.mff.cuni.cz/pdt2.0/>

<sup>4</sup>Swedish Treebank: [http://stp.ling.uu.se/~nivre/swedish\\_treebank/](http://stp.ling.uu.se/~nivre/swedish_treebank/)

```

<seg lang="ro"><s id="31987D0560.n.14.1.ro">
<c></c>
<w lemma="comitet" ana="1+,Ncmsry" chunk="Np#1" head="2:s">Comitetul</w>
<w lemma="cuprinde" ana="1+,Vmip3s" chunk="Vp#1" head="4:vrel">cuprinde</w>
<w lemma="maximum" ana="14+,Rgp" chunk="Vp#1,Ap#1" head="6:num">maximum</w>
<w lemma="douăzeci_și_cinci" ana="1+,Mc" chunk="Np#2" head="2:obj">douăzeci_și_cinci</w>
<w lemma="de" ana="5+,Spsa" chunk="Pp#1" head="6:pcomp-n">de</w>
<w lemma="membru" ana="1+,Ncmp-n" chunk="Pp#1,Np#3" head="2:obj">membri</w>
<c></c>
</s></seg>

```

Figure 2: The treebank annotations.

treebank construction, the Romanian part of the Acquis-Communitaire corpus<sup>5</sup> was used.

Because Romanian is part of the MRLs group, any corpus dedicated to this language must have the sentences tokenized and morpho-syntactically annotated with a tagset that is consistent over the Romanian language. Indeed, all the words of the used treebank are annotated with lemmas, morphosyntactic information (gender, number, person and case) and Part of Speech markers.

The corpus also provide head-word annotation on each constituent. This is precisely the information that has to be encoded in an unlabeled dependency tree, so the dependency structure can simply be extracted from the corpus sentences. An example of the treebank XML-tags for word annotations is given in Figure 2.

### 1.3 Statistical Syntactic Parsers

There are several classes of syntactic parsers that differ upon their output representations: dependency parsing, phrase structure parsing or deep parsing. An advantage of dependency parsing is that dependency trees act as an approximation of the semantics of sentences, and are readily usable in NLP applications. Furthermore, dependency representations have been claimed to be especially well suited for languages with free or flexible word order.

Statistical parsing addresses the problem of constructing a stochastic model to predict the behavior of a random process. In constructing this model, a sample of output from the process needs to be provided based on which predictions about the future behavior of the process could be derived [1].

For data-driven parsers, parsing decisions are made based on learned models. For this reason, usually a parser generator has two stages: the *learning* mode, when a dependency-based treebank of the studied language is provided for the model in order to make it learn (or generate) a parsing model. The second mode is the *parsing* itself, where the model has to assign a well-formed dependency graph for the texts by means of the induced configuration model for the respective language.

MaltParser [8] is a well-known data-driven parser generator, that is language-independent and designed in a modularized fashion being composed of relatively independent

entities. The parameters available in the MaltParser are of three types: *Parsing algorithm parameters*, *Learning algorithm parameters* and *Feature model parameters*. For the current version<sup>6</sup>, MaltParser implements four different families of parsing algorithms, two belonging to the Nivre family and two to the Covington family. Except for the Covington’s non-projective algorithm, the other three algorithms produce only projective dependency graphs.

Based on MaltParser, a data-driven dependency parser [4] for Romanian was developed based on manual annotations of the Romanian Treebank<sup>7</sup> constructed during the RORIC-LING project<sup>8</sup>. Even if the authors report high accuracy scores for the parser (88.6% and 92.0% unlabeled) there are some problems caused by the grammatical coverage of the corpus structures. Another weakness comes from the fact that the used learning model contains mostly the default features of MaltParser and few features are derived from Romanian specific grammatical rules.

The syntactic parser presented in this paper uses a *Maximum Entropy model* [1]. The parser explores a corrective model which recovers heads of non-projective dependency structures by training a classifier to select correct heads of dependency pairs from a set of candidates based on the annotations of the training data sets. When tagging a sentence, the model classifies words from left to right. Each prediction requires a set of context features for each word.

The authors have tested several combinations of features associated with pairs of words, for which the existence or non-existence of a dependency relation has to be predicted by paying attention on the features relevant for Romanian language constructions.

### 1.4 Maximum Entropy Models

The earliest work on discriminative parsing is the local decision Maximum Entropy model of Ratnaparkhi [11], which is trained to maximize the conditional likelihood of each parsing decision within a shift-reduced parsing algorithm. Maximum entropy models have the advantage of dealing with arbitrary, potentially overlapping features over the input sentences, such as dependencies among the sentences’ words. They have been successfully applied to a variety of natural language tasks such as part-of-speech tagging, pars-

<sup>5</sup>This corpus contains about 12,000 Romanian documents of EU laws [3].

<sup>6</sup>The version 1.6.1 of MaltParser is referred here.

<sup>7</sup><http://www.phobos.ro/roric/texts/xml/>

<sup>8</sup><http://phobos.cs.unibuc.ro/roric/>

ing or machine translation [9].

The NLP tasks usually imply statistical classifications that have to estimate the probability of a class  $a$  occurring within the context of another class  $b$ , that will be noted here with  $p(a, b)$ . Contexts in NLP usually include words or data about words. Large corpora usually contain some information about the co-occurrence of  $a$ 's and  $b$ 's but never enough to completely specify  $p(a, b)$  for all possible  $(a, b)$  combinations [11].

To calculate the probability for a word to be a head in a dependency relation, an indicator function is defined as follows:

The *Principle of Maximum Entropy* states that the correct distribution  $p(a, b)$  is that which maximizes entropy or "uncertainty", subject to the constraints which represent "evidence", that is, the facts known to the experimenter ([7]). More precisely, if  $\mathcal{A}$  denotes the set of possible classes, and  $\mathcal{B}$  denotes the set of possible contexts,  $p$  should maximize the entropy:

$$H(p) = - \sum_{x \in \mathcal{E}} p(x) \log p(x) \quad (1)$$

where  $x = (a, b)$ ,  $a \in \mathcal{A}$ ,  $b \in \mathcal{B}$  and  $\mathcal{E} = \mathcal{A} \times \mathcal{B}$ .

In the task of learning syntactic parsing with dependency relations, the fully-labeled data are sentences with words labeled with their heads and dependency relations that connect them with the governor.

## 2. THE PROPOSED METHOD

The mechanism presented in this paper computes unlabeled syntactic tree for sentences in Romanian language. In essence, the model can be considered a language-independent syntactic parser. Indeed, it uses general syntactical and lexical features in training and testing and a set of specific lexical features that come from the Romanian language specific syntactical rules for well-formed sentences (a detailed description of the used features is given in Section 2.2). This set of features can then be changed accordingly to the language the parser is developed.

It is well-known that the parsing models designed for English often focus on learning rigid word order, and they do not take morphological information into account. By means of the proposed mechanism, the authors want to give a solution to the problem of incorporating rich morphology in statistical models.

The model described in this paper is a data-driven syntactic dependency parser that uses a Maximum Entropy statistical mechanism in order to identify the head words of a dependency parsing corresponding to Romanian language texts. A Maximum Entropy model is well-suited for experiments that need to combine diverse forms of context information without imposing any distributional assumptions on the training data [11].

As any other statistical mechanism, the Maximum Entropy mechanism operates in two phrases: the *training phrase* and the *testing or parsing phrase*. During the training process, a set of training instances from the treebank dependency structures are extracted.

For every sentence, the model considers all pairs of words and related to them tries to identify if there could exist a dependency relation between the two words and, with respect to the predicted relation, which word is most likely to be the head of it.

In what follows, the words of a sentence of length  $n$  are identified by their indexes, which, according to the sentence dimension, are elements of the set  $\{1, \dots, n\}$ . In our representation, a word with index 0 has a special meaning, as it represents the virtual root node of the sentence, noted with *Root Node*. Without imposing any restriction, results that the heads for a dependency words index can take values in  $\{0, 1, \dots, n\}$  while the modifiers get values from  $\{1, \dots, n\}$ .

*Definition 1.* Let us consider a sentence of  $n$  words, noted with  $w_1 \dots w_n$ . In what follows, a pair of words  $(w_i, w_j)$  will be named an *ordered pair of words* in one of the following two cases:

- $w_j$  is *Root Node* of the sentence and in this case  $i \in \{1, \dots, n\}$ , or
- $i < j$  for  $i, j \geq 1$ .

An ordered pair of words will be noted with  $(w_i, w_j)_{i < j}$ .

This paper describes a method for constructing an unlabeled dependency syntactic tree, given a sentence of length  $n$  and the morphological and syntactical data for the sentence's words. The mechanism presented in this article is developed in order to predict direction of the dependency arcs in the dependency tree but does not labels them. Attaching labels to the predicted dependency relations should be a matter of a second step, unsupported by the present approach.

Let's consider the sentence  $w_1 \dots w_n$ . The proposed parsing method models the probability function:

$$P : \{1, \dots, n\} \times \{0, 1, \dots, n\} \rightarrow [0, 1]$$

where  $P(i, j)$  represents the probability for the word with the index  $j$  to be head of the word with the index  $i$ ,  $i \neq j$  in the syntactic dependency tree of the considered sentence. This function is based on the Maximum Entropy model trained on the data set described in Section 1.2.

The approach presented here constructs an unlabeled dependency tree with arcs oriented from head to modifier that is rooted on an artificial node, the *Root Node* of the sentence.

The proposed model works with four output values:

$$Output = \{LEFT, RIGHT, ROOT, NONE\}$$

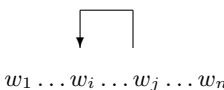
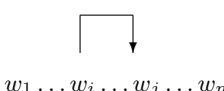
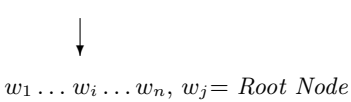
by means of which, the existence or non-existence of a dependency relation between two words of a sentence is encoded.

More precisely, for a sentence  $w_1 \dots w_n$  the output label corresponding to an ordered pair of words  $(w_i, w_j)_{i < j}$  encodes the direction of the dependency relation which is from head to the dependent as follows (see Table 1):

- the output label is *ROOT* if  $w_i$  is a word of the sentence  $1 \leq i \leq n$ , and  $w_j$  is the root node of the sentence ( $j = 0$ );  
this label encodes the case where  $w_i$  does not depend on any other word of the phrase (it is the verb of the phrase or a punctuation mark)
- for  $j \neq 0$ 
  - the output label is *RIGHT* if  $w_i$  is the head of  $w_j$
  - the output label is *LEFT* if  $w_j$  is the head of  $w_i$
- the output label is *NONE* if no dependency relation could be estimated between the words  $w_i$  and  $w_j$

Let us consider the sentence  $w_1 \dots w_n$ . The direction of a dependency arc in the syntactic tree of the sentence between

**Table 1: LEFT, RIGHT and ROOT labels encode direction of a dependency link**

Output label	Direction of dependency link for an ordered pair $(w_i, w_j)_{i < j}$
LEFT	 $w_1 \dots w_i \dots w_j \dots w_n$
RIGHT	 $w_1 \dots w_i \dots w_j \dots w_n$
ROOT	 $w_1 \dots w_i \dots w_n, w_j = \text{Root Node}$

any pair of words  $(w_i, w_j)$ , with  $j \neq i$ , is determined by taking into account some context information concerning the involved words  $w_i$  and  $w_j$ . The contextual data, needed in the training phrase, are the same for both  $(w_i, w_j)$  and  $(w_j, w_i)$  pairs. For this reason, the contextual mapping *Context* used by the model was defined only for ordered pairs  $(w_i, w_j)_{i < j}$  as follows:

$$\text{Context} : \{1, \dots, n\} \times \{0, 2, \dots, n\} \rightarrow 2^{\text{Features}}$$

The function takes a pair of numbers  $(i, j) \in \{1, \dots, n\} \times \{0, 2, \dots, n\}$  such that  $i < j$  for  $j \neq 0$  and returns the context generated by the words  $w_i$  and  $w_j$ . The way the function was constructed ensures that the function maps for any ordered pair of words  $(w_i, w_j)_{i < j}$  the context information of the two words.

## 2.1 The Trained Maximum Entropy Model

By the way it was defined, the model takes into account all the morphological features that could be extracted from the treebank annotations with respect to the two words and their nearby words (both previous and following). These context information are needed in order to predict the existence or non-existence of a dependency link between any ordered pair of words from the sentence. The context data are collected from a context window.

The context window can be of fixed size, calculated according to the position of a particular word, called *target word*, or it can be represented by the entire sentence in which the target word occurs. The dimension of a context window is noted with  $(l, r)$  meaning that it includes  $l$  words to the left and  $r$  words to the right of the target word. Only information in the context window is then used for training classifiers.

The context features algorithm used by the trained model takes each word  $w_i$  with  $1 \leq i \leq n$  and:

- first, the algorithm pairs  $w_i$  with the artificial word noted with *Root Node*.
- then, the algorithm determines ordered pair of words  $(w_i, w_j)_{i < j}$  and pairs  $w_i$  with all  $w_j$  that occur inside

a context window of a certain size related to the word  $w_i$ .

More precisely, the proposed method constructs context windows related to  $w_i$  of a particular size, noted with *Context-Size*<sup>9</sup>. In this case, with respect to the word  $w_i$ , the words  $w_j$  will be chosen inside a context window  $(0, \min(i + \text{Context-Size}, n))$ .

A context is represented by a set of strings where each one is interpreted as a Maximum Entropy feature. If the set of all features used by the model is noted with *Features* then, the context features set related to an ordered pair of words  $(w_i, w_j)_{i < j}$  will be a subset of this feature set.

The trained Maximum Entropy model is defined as the function

$$ME : 2^{\text{Features}} \times \text{Output} \rightarrow [0, 1]$$

which computes the probability of a context,  $\text{Context}(i, j)$ , to be classified with a particular output label. The *ME* function is based on the Maximum Entropy model. Since it takes contexts and classifies them with values from the *Output* set, it requires, as training data, pairs of contexts with known outputs. These training sets are generated during the training phrase of the parser development.

**Remark 1.** The trained Maximum Entropy model satisfies for each ordered pair of words  $(w_i, w_j)_{i < j}$  and for each output label  $o \in \text{Output}$  the following relation:

$$\sum_{o \in \text{Output}} ME(\text{Context}(i, j), o) = 1$$

## 2.2 Feature Patterns

Feature selection plays a crucial role in any Maximum Entropy framework. The general classes of features used in this model for predicting the head words of dependency structures are some of the syntactic features widely used in any syntactic dependency parsing.

<sup>9</sup>The tests involved in this mechanism development had provided better results for context windows of size 8. For this reason, *Context-Size* value will be considered by default to be 8.

**Table 2: The features classes of the model**

Features classes	# of instances of features (1st Data Set)	# of instances of features (2nd Data Set)
Token features	1,926,475	485,429
Context features	2,818,706	639,492
Numeric features	335,980	79,831
Romanian-language specific features	1,380,507	344,358

In order to predict a dependency link between two words just the features over the two words are not enough for high accuracy since all attachment decisions are made outside of the context in which the words occurred. Following the line of the pre-existing statistical models, features of the this parser include the part of speech tags of the words for which context information are determined. Usually, features can represent information which are redundant in the sense that they are represented in multiple different features, in which case we say that the features “overlap”. For this reason, it is better to prefix all features’ values by a string indicating their type to prevent the generation of identical features from different groups of features.

Because Romanian language is a highly inflected language, some morphological features were needed in the model. In this section all the features used in the proposed Romanian dependency-based model are described in detail.

As it was already specified, the context features are determined for each two words of the considered ordered pairs of the parsed sentence. The two words of an ordered pair will be called in the subsequent as *tokens*. The details of the set of features that were used in this model are listed below:

- the fully inflected word forms of the two considered tokens as they appear in the sentence
- the morphologically reduced lemmas of the two tokens together with their morpho-syntactic informations
- quantitative characteristics such as the distance between the two tokens in the sentence (numeric feature)
- the positions of the two tokens in the sentence (numeric feature)
- the part-of-speech (POS) tags of the surrounding words; for each of the two tokens, the words that precedes them (previous two words) and follows them (next two words) in the considered sentence; when the surrounding words are not available (e.g. the tokens are at the beginning or at the end of the sentence) a special feature is used indicating this condition
- POS tag of each intervening word between the two tokens
- the features also checks for definiteness, tense or type agreement information (the gender, number and person agreement if exists) between two tokens  $w_i$  and  $w_j$
- a special feature indicating whether a verb is between the two tokens in the parsed sentence.

The features, even if are treated as strings do not represent only literals but also numeric attributes, like distances or relative positions of words in the sentence.

## 2.3 The Learning Algorithm

The learning algorithm takes as input the training data, which is a parsed set of sentences, manually annotated for syntactic structure and outputs a parsing model. This process of a learning algorithm producing a parsing model from a training set is usually called *training* or *learning*. The parsing model (or the model) contains the parameter settings as well as any feature specifications. We have implemented a left-to-right parsing model in a way that the parser scans through an input sequence from left to right.

The parsing algorithm is implemented in order to determine for a given pair of words the existence or non-existence of a dependency arc in an unlabeled dependency tree, and in case of existence the direction of it. The whole training corpus is preprocessed and each one of the previously described features is extracted.

For each pair of words corresponding to the given sentence, a context is generated using the same contextual function. The output to be learned by the model is deduced from the dependency tree of the sentences used in the training phrase.

Taking into account the contextual function *Context* and the Maximum Entropy model *ME*, the values of the probability function can be determined as in the next equation:

$$P(i, j) = \begin{cases} ME(Context(i, j), ROOT), & \text{if } j = 0 \\ ME(Context(i, j), LEFT), & \text{if } i < j \\ ME(Context(j, i), RIGHT), & \text{if } j < i \end{cases}$$

## 3. EXPERIMENTS

Because Romanian is a morphologically rich language and relatively free word order language, the task of incorporating language specific properties as features in the training model makes it a very important task that influences greatly the overall performance of the parser.

The authors have constantly looked at how different parsing algorithms, feature models and parameter settings influence the parsing accuracy. In this section, the experiments performed during the training and testing of the parser are presented, together with the whole experimental setup: the data sets, the number of used features, the parser evaluations and the corresponding results.

### 3.1 Data Sets

The total treebank sentences were split into a training set and a testing set. The parsing models were trained on approximately 90% of the treebank and we kept out 10% of the corpus for evaluation. The parsing model was trained and tested for words that appear within a context window of a certain size (see Section 2.1), meaning that two words placed at a distance greater than the *Context-Size* value are not taken into consideration.

The evaluation of the parser was implemented in two phrases,

**Table 3: The data sets of the model**

Data Set	# of sentences	# of tokens	Average of tokens per sentence
1st Data Set	1190	24560	20.63
2nd Data Set	717	7674	10.70

**Table 4: The data sets and the corresponding results**

Data Set	Parser Evaluation		
	<i>Recall</i>	<i>Precision</i>	<i>F-measure</i>
1st Data Set	0.80	0.88	0.83
2nd Data Set	0.88	0.93	0.90

that will be identified in what follows upon the used data sets: **1st Data Set** and **2nd Data Set**. **1st Data Set** contains the whole corpus sentences without discriminating sentences based on their length, while the **2nd Data Set** takes into account only sentences including less than 20 words<sup>10</sup>. Table 4 presents some statistics on the accuracy obtained for both data sets. As it was expected, better results are obtained on the **2nd Data Set**, that is, on sentences with less words.

To train and test a syntactical parser requires large quantities of sentences that had to be previously annotated with structural information of the kind the parser has to generate. Unfortunately, annotating real sentences can require a huge amount of work by language experts. The available corpus that was used in the parser development is relatively small (only 1190 sentences). Because of the size of the corpus, the authors chose to evaluate the parser with K- Fold Cross-validation (K=10) which ensures that all the examples in the dataset are used for both training and testing.

Table 2 shows the number of extracted features<sup>11</sup>. The context features are represented by the words and POS data of the previous and following two words corresponding to the tokens of the ordered pairs being studied from the dependency linkage point of view. As it can be seen in Table 2, the Romanian specific features group is comparable with the Token features group, as the model is trained to use an important set of Romanian grammatical information (see Section 2.2).

Experiments were performed using the features described above on a corpus of 1190 sentences, whose parses totaled 607KB training events.

### 3.2 Results

The whole training corpus is processed and each one of the previously described features is extracted. In each iteration, the parse is trained on K-1 folds (9 folds) of data to learn one or more models and then it is asked to make predictions about the head criteria for the remaining validation fold.

The performance of the parser is measured in terms of *Precision*, *Recall*, and *F-measure* such that:

- *Precision* is the fraction of correctly identified/tagged heads

with respect to the number of predicted heads

$$Precision = \frac{No. of correct heads}{No. of predicted heads} \quad (2)$$

- *Recall* is the fraction of correctly identified/tagged heads with respect to the total number of heads of the treebank

$$Recall = \frac{No. of correct heads}{No. of total heads} \quad (3)$$

- *F-measure* is the harmonic mean between *Precision* and *Recall*

$$F-measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (4)$$

For this study, the model's predictions are evaluated based on the annotations of the used treebank which represents the gold standard for our model. The resulted scores for *Precision*, *Recall* and *F-measure* corresponding to both data sets generated for the parser evaluation are given in Table 4. We obtained a precision of 93% for short sentences (maximum 20 words per sentence) and 88% for long sentences (more than 20 words per sentence).

## 4. CONCLUSIONS

A dependency-based syntactic parsing task consists in identifying the head word for each word in an input sentence. Hence, its output is a rooted tree where the nodes are the words in the sentence.

In this paper the training and testing scenario of a dependency-based syntactical parser for Romanian language with statistical models trained on the sentences of a Romanian Dependency Treebank are presented in detail.

The authors consider the obtained results (93% precision for sentences with an average per words of 10 and 88% precision for sentences with words average of 20) are promising, entailing a continuation on the same track with space for improvements. Because any Maximum Entropy model is a data-driven system, a more representative training data is needed in order to ensure the coverage of more syntactical constructs in Romanian language.

## 5. ACKNOWLEDGMENTS

The author M. Colhon has been funded for this research by the strategic grant POSDRU/89/1.5/S/61968, Project ID 61986 (2009), co-financed by the European Social Fund within the Sectorial Operational Program Human Resources Development 2007-2013.

<sup>10</sup>The authors consider that most of the sentences in a natural language do not exceed 20 words. Following this idea, the **2nd Data Set** includes only sentences with maximum 20 words.

<sup>11</sup>The number of features showed in Table 2 correspond to the hole evaluation process, that is, for all 10 iterations of the cross-evaluation.

The author R. Simionescu has been partially funded by the Information and Communication Technologies Policy Support Programme through the project METANET4U - Enhancing the European Linguistic Infrastructure.

The authors gratefully acknowledge the valuable support from Prof. Dr. Dan Cristea.

## 6. REFERENCES

- [1] A. L. Berger, S. A. D. Pietra, and V. J. D. Pietra. A maximum entropy approach to natural language processing. *COMPUTATIONAL LINGUISTICS*, 22:39–71, 1996.
- [2] C. Cherry and D. Lin. A probability model to improve word alignment. In *Proceedings of ACL-2003*, 2003.
- [3] D. Cristea and C. Forăscu. Linguistic resources and technologies for romanian language. *Computer Science Journal of Moldova*, 14(1):34–73, 2006.
- [4] M. Călăcean and J. Nivre. A data-driven dependency parser for romanian. In *Proceedings of TLT-7*, 2009.
- [5] H. J. Fox. Phrasal cohesion and statistical machine. In *Proceedings of EMNLP-2002*, 2002.
- [6] N. Ide, P. Bonhomme, and L. Romary. Xces: An xml-based encoding standard for linguistic corpora. In *Proceedings of the Second International Language Resources and Evaluation Conference. Paris: European Language Resources Association*, 2000.
- [7] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.
- [8] J. Nivre, J. Hall, and J. Nilsson. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC-2006*, pages 2216–2219, 2006.
- [9] F. J. Och and H. Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, 2002.
- [10] C. T. Ogbuji. The future of natural language processing. In *Unix Insider*, 2000.
- [11] A. Ratnaparkhi, J. Reynar, and S. Roukos. A maximum entropy model for prepositional phrase attachment. In *Proceedings of the Human Language Technology Workshop (ARP, 1994)*, pages 250–255, 1994.
- [12] L. Tesnière. *Éléments de syntaxe structurale*. Paris: Klincksieck, 1959.
- [13] C. Watson. Natural language processing and the semantic web. 2005.
- [14] L. Yao, Z. Hao, T. Qian, and X. Wang. Semantic dependency feature selection. In *Proceedings of the 11th Joint Conference on Information Sciences*, 2008.