Ambiguity resolution analysis in incremental parsing of natural language

Fabrizio Costa*, Paolo Frasconi*, Vincenzo Lombardo†, Patrick Sturt‡ and Giovanni Soda*
*Dipartimento di Sistemi e Informatica
Università di Firenze. Italy.
E-mail:{costa,paolo,giovanni}@dsi.unifi.it
†Dipartimento di Informatica
Università di Torino. Italy.
Email: vincenzo@di.unito.it
†Human Communication Research Center
University of Glasgow. UK.
E-mail: patrick@psy.gla.ac.uk

***Abstract*— Incremental parsing gains its importance in natural language processing and psycholinguistics because of its cognitive plausibility. Modeling the associated cognitive data structures, and their dynamics, can lead to a better understanding of the human parser. In earlier work we have introduced a recursive neural network capable of performing syntactic ambiguity resolution in incremental parsing. In this paper we report a systematic analysis of the behavior of the network that allows us to gain important insights about the kind of information that is exploited to resolve different forms of ambiguity. In attachment ambiguities, in which a new phrase can be attached at more than one point in the syntactic left context, we found that learning from examples allows us to predict the location of the attachment point with high accuracy, while the discrimination amongst alternative syntactic structures with the same attachment point is slightly better than making a decision purely based on frequencies. We also introduce several new ideas to enhance the architectural design, obtaining significant improvements of prediction accuracy, up to 25% error reduction on the same dataset used in previous work. Finally we report large scale experiments on the entire Wall Street Journal section of the Penn Treebank. The best prediction accuracy of the model on this large dataset is 87.6%, a relative error reduction larger than 50% compared to previous results.**

***Index Terms*— Recursive neural networks, structured data, first pass attachment, incremental parsing, learning preferences.**

I. Introduction

The incremental strategy is a largely accepted hypothesis about the human mechanism of syntactic processing in language comprehension. Under this model, processing proceeds from left to right, and each input word is assigned a structural position as it is being read [1]. The incrementality hypothesis is supported by several experimental studies that demonstrate how humans are able to assign a meaning to "almost any" initial (left) fragment of a sentence [2], that is, they are capable of anticipating syntactic and semantic decisions before reaching the end of the sentence [3]–[5]. In particular, under the *strong incrementality* framework (assumed in this paper), humans maintain a totally connected parse tree while scanning the input words from left to right, with no input stored in a disconnected state [6]. Although well accepted in the psycholinguistic community, incremental processing has received relatively modest attention in the computational linguistic community. In this direction, Roark & Johnson [7] have proposed a top-down left-corner probabilistic parser that uses a probabilistic best-first strategy and a beam-search heuristic to avoid the non termination problems typical of top-down predictive parsers. Their parser proceeds incrementally from left to right, with one item of look-ahead, and maintains a fully connected tree spans the left context and is used to extract non-local dependency information. With regards to connectionist architectures, Lane & Henderson [8] have proposed Simple Synchrony Networks and applied them to a small scale parsing problem. Their approach combines Temporal Synchrony Variable Binding with Simple Recurrent Networks in order to output representations of tree structures. More recent work by Henderson [9] has used neural networks to estimate the parameters of a generative model, resulting in a wide-coverage statistical parser with state-of-the-art performance. Earlier connectionist parsing models tackling smaller-scale problems have included those of Jain [10], Kemke [11], Miikkulainen [12] and Wermter & Weber [13].

In this paper we focus on the development of machine learning methods for ambiguity resolution in first-pass attachment. By "first-pass attachment", we mean the process of initially combining a new input word with the developing phrase structure analysis of a sentence. This process can be viewed as one component of a full parsing model, in which we would also need to implement a way of keeping track of alternative analysis, and recovering them if necessary. The first-pass attachment problem is mainly important in psycholinguistics, where much of the experimental research involves testing initial preferences for ambiguity resolution during the processing of a written or auditorily presented sentence. However, a solution to the first-pass attachment problem could help in building computational tools whose behavior is closer to that of humans in ambiguous situations. To gain some intuition on the ambiguity resolution problem

and why statistical regularities may play an important role, let us consider the ambiguous sentence "the servant of the actress who was on the balcony died". Cuetos & Mitchell [14] report evidence that English and Spanish speakers have a different preferential bias in attaching the ambiguous relative clause. In particular, English speakers are more likely to support the interpretation in which the actress was on the balcony (attaching the clause to the most recent noun), while for the Spanish translation of the sentence native speakers are more likely to support the interpretation in which the servant was on the balcony (attaching the clause to the less recent noun). Mitchell et al. [15] showed that preferences for this type of ambiguity could be modulated through exposure, and proposed that the difference can be explained as a consequence of different structural frequencies in the two languages, independently of lexical preferences. They have formulated the *tuning hypothesis*, according to which purely structural statistical regularities determine the earliest stages of syntactic ambiguity resolution in human parsing.

In [16] we have proposed a computational model in the attempt of verifying the above hypotheses with the help of machine learning. As revised in Sections II and III, our method is based on a dynamic grammar as a model of strong incremental parsing. States of the dynamic grammar consist of incremental trees, i.e. the substructures T_k of a parse tree T that span the first words $w_1, w_2, \ldots, w_k$ in a sentence. The graph difference between two consecutive incremental trees T_k and T_{k-1} is called in this framework a *connection path* and can be seen as the syntactic structure that must be added to the incremental tree spanning $w_1, \ldots, w_{k-1}$ in order to attach the next word w_k. Transitions in this grammar are generated by extracting a set of connection paths from a treebank (using the algorithm proposed in [17]). Under this model, ambiguity resolution consists of choosing the correct transition to be applied at each step in order to continue parsing. The problem can be conveniently modeled as a preference learning task in which instances consist of alternative incremental trees (each representing a valid continuation). We propose a recursive neural network (RNN) [18], [19] to learn this preference task[1]. Results in [16] support in a quantitative way the psycholinguistic hypotheses that structural learning plays a significant role for disambiguation. In particular, we found that the RNN trained on a relatively large collection of parse trees (extracted from the Penn treebank [21]) was actually able to reproduce some interesting patterns of human syntactic disambiguation on new sentences. There have been very few other psycholinguistic models of parsing that combine connectionist methods with symbolic representations of syntactic structure in this way. Some exceptions are the hybrid models proposed by Stevenson [22] and Vosse & Kempen [23], both involving the creation of the syntactic structure through a network-based process of competitive activation. However, unlike [16], these models do not employ connectionist learning techniques, and are not designed to be used in a wide-coverage setting.

[1]In [20] Collins and Duffy proposed a kernel based approach for solving a related preference learning problem over sets of syntactic trees; in their paper alternatives consist of complete parse trees for a sentence that are given a high score by a statistical parser.

In this paper we report about new results obtained on a significantly larger data set than that used in [16], and we present a thorough analysis of the properties of the trained network. After performing several statistical tests that correlate structural features of the input to the generalization error, we find that the learned solution consistently assigns higher scores to simpler and more frequent structures. Moreover we find that relevant signals tend to concentrate near the anchoring point between an incremental tree and a connection path. Interestingly, this observed behavior can be exploited to improve the design of the predictor by selectively pruning nodes that are too distant from the candidate attachment points. Selection of relevant portions of a tree when learning in structured domains can be seen as a counterpart of attribute selection for attribute-value data. Here substructure selection is driven partly from domain knowledge and partly from the analysis of prediction errors. Domain partitioning is a second technique that we find useful to inject prior knowledge and to boost prediction accuracy. More precisely, we propose to specialize separate networks on different domain splits, according to the grammatical category of the word to be attached. All these enhancements produce a significant accuracy improvement over the previous architecture.

The rest of the paper is organized as follows. In Section II we review the incremental dynamic grammar and we formulate disambiguation as a preference learning task. In Section III we briefly revise RNNs for learning preferences on syntactic structures. In Section IV we describe and characterize the data set. In Section V we study the main properties of the trained network and we correlate prediction error to the structural properties of the input trees. In Section VI we describe the enhanced architecture and report the wide coverage experiments.

II. The incremental dynamic grammar model

In this section we give some basic concepts related to first-pass ambiguity resolution. More details can be found in [17].

A. Definitions

We assume that syntactic processing takes place in a strongly incremental fashion. This means that each word w_i is processed by scanning the sentence from left to right and that we do not allow disconnected sub-structures to be assembled together at some later stage in processing.

Let T be the parse tree for sentence $w_1, \ldots, w_n$. We define for each $i = 1, \ldots, n$ the *incremental tree* T_i as the sub-tree of T recursively built in the following way (see Fig.1 (a)):[2]

- T_1 consists of the chain of nodes and edges of T that goes from w_1 to its maximal projection[3].
- T_i consists of all the nodes and edges in T_{i-1} and either:

[2]The examples of syntactic structure in this paper are based on the actual structures used to train the network (a relatively flat representation derived from the Penn Treebank format [21]).

[3]A maximal projection of a word w is the largest non terminal symbol X that is related to w through projection (i.e. w and X share head features). For example, a noun projects onto a Noun Phrase.

- the chain of nodes and edges of T descending from node R where R is the lowest node of T_{i-1} that dominates w_i
- the chain of nodes and edges of T descending from node R where R is lowest node of T that dominates both the root of T_{i-1} and w_i, and the chain of nodes and edges that connect R with the root of T_{i-1}

Given two incremental trees with indices $a < b$, T_a and T_b, we define the *difference* between T_b and T_a as the set of all the edges that are in T_b but not in T_a and all the nodes touched by those edges. The difference between T_i and T_{i-1} is called the *connection path*: $cp_i = T_i - T_{i-1}$, (see Fig.1 (b)). The node that belongs to T_{i-1} and cp_i is called the *anchor* (this is the node where cp_i attaches to T_{i-1}). The preterminal node for word w_i is called the *foot*. This is the node whose label is the "part of speech" (POS) tag of word w_i and in our framework it is a leaf of the syntactic tree. POS-tags are the syntactic category of words and can be predicted with very high accuracy [24]. We use the symbol '$\circ$' to denote the *join*

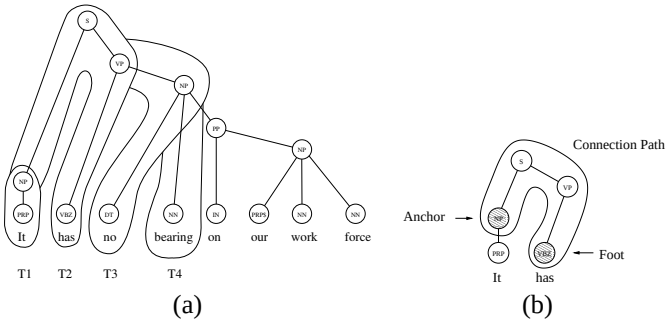


Fig. 1. (a) Example of incremental trees. (b) Anchor and Foot nodes.

operator, defined as $T_i = T_{i-1} \circ cp_i$. According to the above definitions, an incremental tree T_i can always be written as the result of a number of joins: $T_i = cp_1 \circ cp_2 \ldots \circ cp_i$.

We assume that the children of each node are ordered from left to right. The *right frontier* of an incremental tree is the chain of the current rightmost children, starting from the root node and ending to the rightmost leaf (see Fig.7). Join operations are always performed on nodes belonging to the right frontier.

Lombardo and Sturt [17] describe a procedure that takes as input a parse tree T and computes all the incremental trees $T_1, \ldots, T_n$ and the all the connection paths $cp_1 \ldots cp_n$. By applying this procedure to a treebank B (a set of sentences annotated with their parse trees) we obtain a set of connection paths called the *universe of connection paths*, denoted $U(B)$.

B. First Pass Attachment Prediction

Suppose we are given a new sentence $w_1, \ldots, w_n$ not included in the treebank B, and suppose that at stage i of parsing we know the correct incremental tree T_{i-1} spanning $w_1, \ldots, w_{i-1}$. We want to compute the next tree T_i in order to accommodate the next word w_i. Under the implicit hypothesis that $U(B)$ contains the required connection path, T_i can be obtained by joining T_{i-1} to some unknown path cp^* in $U(B)$. The prediction problem is then defined as follows: given T_{i-1}, find $cp^* \in U(B)$ such that $T_{i-1} \circ cp^*$ is the correct incremental tree spanning $w_1, \ldots w_i$. The set of candidate paths can be significantly reduced by enforcing the following two rules that must be satisfied by a legal joining:

- the foot of cp^* must match the POS-tag of w_i;
- the anchor of cp^* must match the one of the nodes in the right frontier of T_{i-1}.

Note that $U(B)$ along with the joining operator and the above rules can be regarded as a dynamic grammar [25] [26]. This grammar, however, is highly ambiguous as the set of connection paths that satisfy the two joining rules may be very large. In particular, there are three different sources of ambiguity:

- a word can have more than one POS tag.
- the anchor can be any node of the right frontier (see Fig.2 (a)).
- for each pair <anchor tag, foot tag> there can exist more than one legal connection path (see Fig.2 (b)).

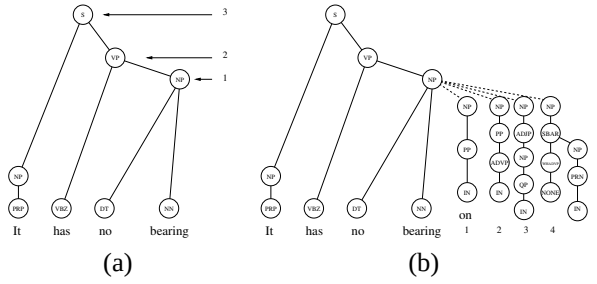


Fig. 2. (a) Anchor variability. (b) Connection path variability.

The set of trees obtained by legally joining T_{i-1} to a path in $U(B)$ will be referred to as the *forest of candidates* for word w_i, denoted $F_i = (T_{i,1}, \ldots, T_{i,k_i})$. Note that, under our assumptions, one and only one tree in F_i is the correct incremental tree spanning $w_1, \ldots, w_i$. Without any loss in generality we will assume that the first element in the forest is the correct one.

C. Left Recursion and Lexical Information

The top–down parser suffers from the problem of left recursion [27]. Since a left recursive structure can be arbitrarily nested, we cannot predict the correct connection-path incrementally. There are a few practical and psycholinguistically motivated solutions in the literature [28], but in the current work we have resorted to an immediate approach which is extensively implemented in the Penn Treebank schema: that is we *flatten* the tree structure and avoid the left recursion issue altogether. Consider as an example the application of the flattening procedure to a tree like 1) that produces as a result a tree like 2):

1) [$_{NP}$ [$_{NP}$ DT NN] PP]
2) [$_{NP}$ DT NN PP]

Since the main focus of the present linguistic analysis is about syntax, no lexical information is used for prediction and therefore two sentences having the same sequence of POS tags are equivalent in our study. We believe that the use of lexical information would further improve the prediction

capability of the model, although this would result in a more complex network architecture. However, the current network architecture allows us to model theories in which purely structural information plays the major role in first-pass ambiguity resolution, such as the Tuning Hypothesis [15].

D. Connectionist vs. frequency approach

According to the formulation given above, we can restate our learning task as the estimation of a utility function that, given a forest of incremental trees $F_i = (T_{i,1}, \ldots, T_{i,k_i})$, computes the highest value for the correct element $T_{1,i}$. A first and direct approach is to derive a probabilistic estimator of such function by collecting information on occurrences of all the instances of our problem (distinct trees) in a large corpus. We want to estimate $P(T_i) = P(T_{i-1}, cp_r)$, that is $P(cp_r|T_{i-1})P(T_{i-1})$. This approach suffers from a severe data sparseness problem. The combinatorial nature of the grammar determines negligible probabilities for the occurrence of the incremental trees in training sets of any given size (i.e. $10^4 - 10^6$ sentences) currently available. To quantify this statement, we have selected a sample of 1,000 sentences randomly divided in two sets of same size: one for a nominal test set and one for a nominal training set. We have calculated the number of trees of the test set present in the training set, counting first the coincidences among correct incremental trees, and then among all trees (i.e. the correct incremental trees plus all incorrect trees generated by the dynamic grammar), obtaining for the correct incremental trees:

Correct trees in test set	11,011 trees
Correct trees in training set	11,250 trees
Correct trees from test set in training set:	420 trees
Percentage	4%

and for the overall dataset (i.e. correct incremental trees plus all incorrect trees generated by the dynamic grammar):

Trees in test set:	480,928 trees
Trees in training set:	517,308 trees
Trees from test set in training set:	4,469
Percentage	1%

The small percentage of the training set seen in the test set clearly illustrates the infeasibility of a direct multinomial estimator.

In computational linguistics, data sparseness is traditionally dealt with smoothing techniques, that is, by approximating frequency estimation through the decomposition of complex and infrequent objects in more frequent sub-parts [29]. The item probability is computed by composing the frequencies of the sub-parts, under some independency hypothesis. In an incremental framework, this decomposition has been attempted by [7]. Our solution does not make any simplifying assumptions, and tries to take advantage of the global information available, overcoming at the same time the data sparseness problem. This is achieved by resorting to a parametric estimator (the RNN) that makes use of a much smaller set of hyper-parameters. This can be viewed as a way to perform an adaptive compression of the information.

III. Recursive networks for preference learning

We present a two-step solution to the first-pass attachment prediction problem. In the first step, RNNs are used to adaptively build a set of features that describe a parse tree as a fixed-size real vector. In the second step we show a utility function solution to the preference learning task.

A. Recursive neural networks

The general theory developed in [18] allows the processing of directed acyclic graphs with a super-source. Here we are interested in the case of labeled ordered q-ary trees. By ordered we mean that, for each vertex v, a total order is defined on the q children of v. $L(v)$ denotes the label attached to vertex v of T. In the case of syntactic trees, labels belong to a finite alphabet of nonterminal symbols $\mathcal{N} = \{\alpha_1, \ldots, \alpha_N\}$. The set of all trees with labels in $\mathcal{N}$ is denoted as $\mathcal{T}^{\#}$.

The basic neural network architecture computes a vector of n features according to the following recursive processing scheme:

$$\begin{aligned} \boldsymbol{\varphi}(\mathit{nil}, T) &= 0 \\ \boldsymbol{\varphi}(v, T) &= \tau(\boldsymbol{\varphi}(u_1, T), \ldots, \boldsymbol{\varphi}(u_q, T), L(v)) \end{aligned} \tag{1}$$

where u_r, $r = 1, \ldots, q$ denotes the (possibly missing) r-th child of v. We can interpret the above equation as the recursive state space representation of a generalized dynamical system that "evolves" on a tree domain [18]. Under this interpretation, the feature vector $\boldsymbol{\varphi}(v, T) \in I\!R^n$ is a *state* vector associated with node v of tree T, $\tau : I\!R^{q \cdot n} \times \mathcal{T} \mapsto I\!R^n$ is the state transition function that maps states at v's children and the label at v into the n-dimensional state vector at v. If a child is missing, the corresponding argument to τ is the *frontier* state $\boldsymbol{\varphi}(\text{nil}, T) = 0$. States in Eq. (1) are updated bottom-up, yielding a vector based representation $\boldsymbol{\phi}(T) = \boldsymbol{\varphi}(\text{root}[T], T)$ at the root of T that can be seen as the result of applying a feature mapping to the entire tree. Using a parameterized function τ (e.g. realized by a feed-forward neural network) this feature mapping can be made adaptive. In the following we call state transition network the network implementing the mapping τ.

B. Preference learning

Typical prediction problems are formulated either as classification or as regression, which can both be thought of as function approximation problems, where the image of the function is a subset of $I\!R$ for regression or a discrete set for classification. Ranking a set of alternatives is a task with characteristics of both the previous problems: like classification, it has as its image a discrete set, and like in regression, there exists an ordering relation among the elements of the image. Ranking problems have been studied under different assumptions in machine learning [30] [31]. Here we are interested in the simple case of preference learning, in which data points are organized into sets of instances and *exactly one* instance is preferred to the rest in the set. Without losing generality we can write a preference data set as a collection of (partially ordered) sequences

$$\mathcal{F}^m = \{(T_{i,1}, T_{i,2}, \ldots, T_{i,k_i})\}_{i=1}^m$$

where $T_{i,j} \in \mathcal{T}^{\#}$, k_i is the size of the i-th set and, conventionally, $T_{i,1}$ denotes the preferred instance in its set.

In order to solve the preference learning problem we use the *utility function* approach in which we learn a function

$$f : \mathcal{T}^{\#} \mapsto I\!R$$

and, for a future forest of trees $(T_1, \ldots, T_k)$ we predict that the preferred instance is T_j iff

$$j = \arg \max_{l=1,\ldots,k} \{f(T_l)\}.$$

We propose realizing f in the following way:

$$f(T_l) = \boldsymbol{w}^T \boldsymbol{\phi}(T_l) + b \tag{2}$$

where $\boldsymbol{w} \in I\!R^n$ and $b \in I\!R$ are adjustable weights and $\boldsymbol{\phi}(T_l)$ is the feature vector of T_l. The utility function f can be then used to compute the probability of selecting the correct tree:

$$p(Y = j) = \frac{e^{f(T_j)}}{\sum_{l=1}^{m} e^{f(T_l)}}$$

and, according to the maximum likelihood principle, we can learn f by maximizing the objective function

$$E = -\sum_{i=1}^{m} \log p(Y = 1) = -\sum_{i=1}^{m} \log \frac{e^{f(T_{i,1})}}{\sum_{l=1}^{m} e^{f(T_{i,l})}} \tag{3}$$

jointly with respect to the parameters $w_1, \ldots, w_n, b$ of Eq. 2 and the parameters of the network implementing the state transition function of Eq. 1. Maximization can be carried out by gradient descent as explained in the next subsection.

C. Parameter Optimization

Gradients are computed by a special form of back-propagation on the feed-forward network obtained by unrolling the state transition network according to the topology of the input tree as showed in section III-A. The algorithm was first proposed in [32] and is referred to as *back-propagation through structure* (BPTS). Backward propagation proceeds from the root to the leaves. Note that gradient contributions must be summed over all the replicas of the transition network to correctly implement weight sharing. In Fig.3 we depict the coupling and unfolding of the transition network and the output network on a forest of two incremental trees.

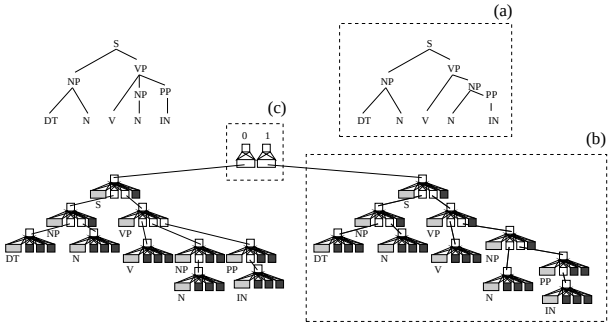


Fig. 3. Network unfolding on a forest of two elements. (a) Syntactic tree. (b) Unfolded recursive net. (c) Utility function network.

IV. Linguistic Corpus and Network Training

A. Linguistic Corpus

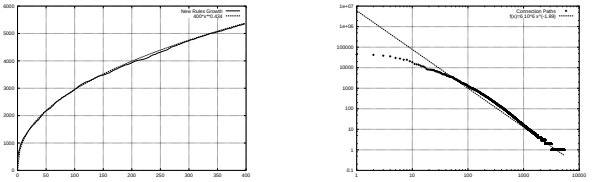
All the experiments in this paper are based on the Wall Street Section of the Penn-Treebank Corpus [21]. We have adopted the standard setting widely accepted in the literature (see [33]): specifically, sections 2-21 have been used to form the training set (39,832 sentences, 950,026 words), section 23 has been used for the test set (2416 sentences, 56,683 words) and section 24 for the validation set (3,677 sentences, 85,335 words). The entire dataset used for our experiments includes therefore 45,925 sentences for a total of 1,092,044 words. The average sentence length is 24 in a range of 1-141 (1-67 in the test set). The labels (tags) on the nodes of the parse trees can be divided into Part-Of-Speech (POS or pre-terminal) tags, and non-terminal tags: POS tags dominate a single lexical item and indicate the syntactic category of the item (ex. a noun or a verb), while non terminal nodes dominate sequences called "phrases" that can be made of pre-terminal and/or non-terminal tags. In the Penn Treebank the POS tags are 45, and the non-terminal tags are 26. Although the syntactic annotation schema provides a wide range of semantic and coindexing information, we have used only syntactic information about the dominance relation[4].

B. Connection Path Analysis

One of our working hypothesis is the "coverage assumption", which states that we can extract, from a large corpus, the complete set of connection paths with which to form all possible incremental trees. This is likely to be only approximately true, and we perform the following experiment to get a quantitative result on the validity of this assumption. We build sub-sets with an increasing number of sentences: from 100 to the full 40K sentences in steps of 100 sentences. The list of connection paths with their frequencies, and the number y of distinct connection paths were then extracted from each sub-set by simulating an incremental parse of each sentence. The simulator took as input the parse tree for a sentence and, scanning each word from left to right, marked the subgraph of the tree that connected the new word to the previous incremental tree as a connection path (see [17] for details). Fitting the results (see Fig.4 (a)) with a polynomial model we have $y = ax^{\alpha}$ where $\alpha = 0.434$. This has a remarkable similarity with Heaps law [35], an empirical rule which describes the vocabulary growth as a function of the text size. Heaps law establishes that a text of n words has a vocabulary of size $O(n^{\beta})$ with $\beta \in [0, 1]$ where for English in first approximation $\beta = \frac{1}{2}$. Considering how often certain connection paths were used in the making of the syntactic trees we observed that the frequency counts distribute accordingly to another famous linguistic law that bears the name of Zipf's law. Zipf's law expresses a relationship between the frequency of words occurring in a large corpus and their rank. Given a corpus under examination, the rank of a word is defined as the position of that word in the list of all the words in

[4]This limitation can be a very compelling one, since most parsing models achieve valuable results by including lexical statistics on word occurrence and functional dependencies ([34], [29]).

the corpus, ordered by frequency of occurrence. According to Zipf's law $f \propto \frac{1}{r}$, which can be viewed as the existence of a constant k such that $f \cdot r = k$. What the law states is that there are few very common words and many low frequency words. Considering connection paths instead of words we found that their frequency was closely described by the Zipf's law (Fig.4(b)). Moreover we found that the same distribution



(a) (b)

Fig. 4. (a) Number of different connection path in respect of dataset size (units in 100 sentences). (b) Connection paths Zipfian distribution.

holds if we keep distinct the connection paths whose foot node belongs to a specific category (such as nouns, verbs, articles, ...). Finally we saw that new connection paths extracted after having processed 10,000 sentences were rarely used (less than 1% of the time) in successive trees. This supported our hypothesis on the coverage approximation.

C. Network Training

For each word w_i in a sentence, a forest of alternatives was generated by extracting the incremental tree T_{i-1} spanning $w_1, \ldots, w_{i-1}$, and joining it with all the legal connection paths. Each sentence had an average length of 24 words and each forest contained on average 120 trees. Considering that the average number of nodes of an incremental tree was 27, we have that the entire dataset had $1 \cdot 10^6$ forests, $117 \cdot 10^6$ trees, and $3 \cdot 10^9$ nodes.

The learning regime is online: weights are updated after the presentation of each forest. A separate set of 1,000 sentences from section 24 of the treebank was used as a validation set to control overfitting by early stopping. Given the considerable amount of training data, accuracy on the validation set was monitored after the presentation of each group of 100 sentences. Optimization was stopped if the validation error reached a minimum and did not decrease for the subsequent 1,000 sentences. In this setting, three epochs were enough to reach the optimum.

The state transition network was implemented as a single layer feed-forward network with an input vector composed by 1 unit for the threshold, 71 units for the one-hot encoding of the non-terminal and POS tag symbols and 25 units to represent the state vector of each child node. We noted that the longest production in the dataset had 51 nonterminal symbols on its right hand side, and that productions with more than 15 nonterminal symbols were very rare. Since each nonterminal position was associated with its own set of weights in the transition network, we pruned long productions in order to avoid poor estimates of the weights associated with positions that are infrequently filled in. Pruning beyond the 15th position resulted in a reduction of only 0.3% of all the productions. The input layer had therefore a total of 447 units. The output layer (which encoded the state of the node being processed) was made of 25 units. The state transition network had a total of 11K free parameters. The non-linearity was the standard hyperbolic tangent function. The utility network was a feed-forward network with 25+1 units in input and 1 unit in output. Once the recursive network was unrolled, the forward and backward phase proceeded following the standard back-propagation algorithm with a fixed λ learning rate and momentum m. Good values for the parameters λ and m have been experimentally determined on a working set to be $\lambda = 10^{-3}$ and $m = 0.1$. Training the system on the whole dataset took less than 3 days of CPU per epoch on a 1GHz Pentium III Intel processor.

We evaluated the learning curve of the system. The training set has been partitioned into sub-sets of 100, 400, 1,000, 4,000, 10,000 and 40,000 sentences. A validation set of 1,000 sentences was used for early-stopping. We report in Fig.5 the percentage of times that the correct element has been ranked by the system in the first position on a test set of 2,416 sentences. On the x axis we report the number of training sequences and on y the fraction of training trees correctly ranked in first position. The results indicate that the difference between training with 10,000 or 40,000 sentences yields a 3% relative error reduction.

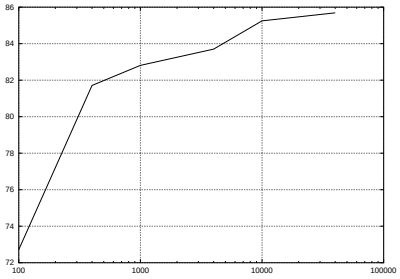


Fig. 5. Learning curve for the experiment described in Section IV-C

V. Structural analysis

In this section we will characterize the preferences learned by the RNN trained on a large corpus. We will start by analyzing the correlation between the network's performance and the structural properties of the incremental trees. Then we will study the influence of the frequency of the connection paths on the choices of the system. Finally, we will compare the preferences learned by the network with some heuristics studied in psycholinguistics.

A. Correlation between structural features and accuracy

The aim of conducting the analysis reported here is to uncover those structural properties on which the network makes its decisions. Our approach is to test hypotheses about these structural features, and to test them via statistical analysis of the network's performance on a sample of test items. The results will be informative for a general understanding of the

network architecture, and can be used to refine the system to increase performance.

The set of structural features under investigation is reported in Table I. These features include both complexity measures of the trees, and statistical properties of the data set. They can be grouped into the following sets:

- number of nodes in the incremental tree (rows: nodes in tree, nodes in cpath, max outdeg, anchor outdeg, root outdeg): a higher value implies a more difficult error assignment task for the network when propagating the error. Moreover the tree automaton that we approximate with the connectionist system is more complex as the number of possible configurations increases;
- height of the incremental tree (rows: anchor depth, tree height, cpath height): a greater height implies more steps in the propagation of the information and as a consequence a weakening of the gradient information;
- frequency of the connection path (row: freq of cpath): since after all the RNN is a statistical estimator, this count is a valuable baseline for comparing the RNN performance.

In addition, we study the number of alternative incremental trees (row: forest size), as we expect a negative correlation between the number of alternatives and the prediction performance. Finally, we study the word absolute position (row: word index), since words that occur later in the sentence condition the choice of the correct attachment on a more complex and variable context.

To summarize, we hypothesize that the degree of error in the network's assignment of preference to an incremental tree (i.e. $1 - p$) will correlate positively with the structural complexity of that tree. We also hypothesize that the error will correlate negatively with the frequency of the connection path.

For each of the features of interest we have collected basic statistical information (max value, mean, standard deviation, skew, kurtosis) and tested for normality so to be able to use the appropriate statistical test later on. To evaluate our hypotheses, we computed the Spearman's correlation coefficient between these features and the network's error on the correct element. The correlation was run separately for each feature, over a randomly sampled sub-set of 200 pairs (error, feature). We report the correlation coefficients (column: ρ) in table I. The test indicates a significant, if small, positive correlation between each feature and the network's error, except the root outdegree, and a relevant negative correlation between the frequency of the connection path and the error. The most significant positive correlations are with the size of the connection path and the forest size. Thus the correlation results support our hypotheses.

Description	max	mean	std dev	skew	kurt	ρ
max outdeg	18	4.3	1.9	0.4	4.8	0.18*
tree height	28	6.6	3.9	0.7	3.8	0.19*
nodes in tree	122	27.3	19.2	0.8	3.4	0.20*
cpath height	5	1.5	0.7	0.4	3.4	0.34*
nodes in cpath	11	2.7	1.0	1.6	7.5	0.33*
anchor outdeg	18	2.6	1.6	0.8	7.2	0.21*
anchor depth	28	4.6	3.9	1.0	4.4	0.17*
root outdeg	18	2.9	1.6	1.2	6.5	0.02^{ns}
forest size	1940	126.3	145.1	2.4	11.9	0.31*
word index	66	14.7	10.4	0.9	3.6	0.19*
freq of cpath	102291	177.9	2077.6	30.1	1226.1	-0.39*

TABLE I

STATISTICS FOR THE 11 FEATURES USED IN THE STUDY OF THE TRAINED NETWORK BEHAVIOR. THE FINAL COLUMN (ρ) INDICATES THE SPEARMAN'S CORRELATION COEFFICIENT BETWEEN THE RELEVANT FEATURE AND THE NETWORK'S ERROR. ($* : p < .05$; NS: NOT SIGNIFICANT)

B. Structured characterization of true and false positives

In the next set of analysis, we investigate the hypothesis that the network learns to prefer simpler structures to complex structures, and that this preference influences its decision, both when it identifies the correct incremental tree, and when it mistakenly chooses an incorrect incremental tree. To evaluate this hypothesis, we distinguish *true positive* elements and *false positive* elements: true positives are the correct incremental trees that are preferred by the network; false positives are the trees preferred by the network but that do not correspond to the correct trees in the treebank.

At first we will identify statistically significant differences in the average values of some features. Then, analyzing the distinctive features, we will identify some characterizing properties of the set of true positive elements against the second preferred element, and we will do the same with the false positive against the correct elements. For the features that do not exhibit a normal distribution we use the Wilcoxon Matched-Pairs Signed-Ranks Test on a random sample of 200 pairs from the dataset for each feature. For all the other features a paired t-test is used, randomly sampling 100 pairs from the dataset for each feature.

In the first experiment the tests are used to determine whether there are meaningful differences in some feature of the trees when comparing the network false positive with the correct elements. These tests are informative about the impact that the structural features have on the network incorrect choices. We report the results that are significant ($p < .05$) under the respective statistical tests in Table II. In column "correct elem" we report the average values for the correct element that the RNN has not been able to predict and in "false pos" column we report the average values for the wrong element picked by the net. In the last column, we report the size of the difference Δ, stated in terms of the standard deviation of the corresponding distribution. The interesting

Description	correct elem	false pos	Δ / sd
tree height	7.38	7.20	0.05
# of nodes in tree	30.91	30.55	0.02
cpath height	2.12	1.67	0.64
# nodes in cpath	3.35	2.77	0.58

TABLE II

MEANS FOR STRUCTURAL CHARACTERISTICS FOR THE CORRECT ELEMENT VS. THE ELEMENT INCORRECTLY CHOSEN BY THE NETWORK. TABLE SHOWS FEATURES WHERE PAIRWISE COMPARISONS WERE SIGNIFICANT AT $p < .05$.

result of this experiment is that in the case of wrong predictions, attachment choices predicted by the trained RNN yield, on average, oversimplified trees. As shown in Table II, the value of each of the four statistically significant features is smaller in the incremental tree obtained by following the RNN's predicted attachment than the value taken by the same feature on the correct incremental tree. This indicates that the network preference for simpler trees has a measurable impact on performance: the network has a significant tendency to choose the incorrect tree because it is simpler than the correct alternative.

There was no significant effect of the outdegree on the RNN false positive error. We note how the differences within the whole incremental tree are much smaller than those between individual connection paths. This indicates that connection paths are the key element responsible for the discrimination between the correct element and the incorrectly chosen element. We will be using this finding for enhancing the performance of the system.

In a second experiment we tested differences between true positives and the element ranked second by the net. This allows us to determine the information on which the network bases its preference when it correctly predicts the appropriate element. We report the significant results in Table III (to be read as the previous one). The same trend was kept for the true positives:

Description	true pos	second elem	Δ / sd
tree height	6.02	6.56	0.14
# of nodes in tree	24.08	25.32	0.06
cpath height	1.47	1.81	0.49
# nodes in cpath	2.56	3.08	0.52
anchor outdegree	2.48	2.72	0.15
root outdegree	2.8	2.98	0.11

TABLE III

MEANS FOR STRUCTURAL CHARACTERISTICS FOR THE CORRECTLY CHOSEN ELEMENT VS. THE ELEMENT RANKED SECOND BY THE NETWORK. TABLE SHOWS FEATURES WHERE PAIRWISE COMPARISONS WERE SIGNIFICANT AT $p < .05$.

the RNN has preferred the correct incremental trees because of their "simplicity" in comparison to the second ranked alternative, which turns out to be more complex. Note that now the root and the anchor outdegree have become a meaningful feature in a way that is still consistent with the hypothesis that "simpler" trees are preferred, i.e. the correct incremental trees have roots and anchors with smaller outdegrees. This latter fact can be represented by a heuristic that disprefers joining those connection paths that increase the number of children of the root or of the anchor, since this leads to wrong incremental trees.

We suspect that the simplicity preference of the network is mainly due to the combinatorial nature of the elements of this domain, since all the features are strongly correlated, and there could be an underlying factor that is the direct cause of the preference. Analyzing the Zipfian distribution of connection paths we find that shorter connection paths are more frequent. As a direct consequence, most correct incremental trees are themselves simpler because they are more frequently derived by joining simpler elements. In order to understand the magnitude of this effect we have run a Pearson Correlation test on a sample of 10000 pairs of connection paths number of nodes vs. log(freq). We obtain a correlation of $\rho = -0.33$ (statistical significance $p < 0.001$) indicating that smaller connection paths are reliably more frequent.

In the following subsection, we therefore investigate the influence of connection paths frequencies on false positives and true positives, respectively.

C. Influence of connection paths frequencies

In Fig. 6 there are several comparison between the network results and other psycholinguistic or frequency–based preferences. Here we compare the RNN to the simple frequency heuristic obtained by ranking each alternative connection path according to its corpus frequency. The test is done on the standard test set of 2416 sentences (Section 23). Each point (x,y) in the diagram of Fig. 6 is to be interpreted as: y is the proportion of times that the correct element has been ranked in the position x or less. From Fig.6 we can deduce that the RNN bases its decisions on something more than the pure frequency.

A paired t-test was used to determine the influence of the log-transformed frequency[5] of the connection path on the network accuracy. As in the previous analysis, pairwise comparisons were conducted both for true positives and for the false positives. For the true positive, the mean log-frequency of the connection path was 9.2 against a mean of 5.2 for the second best ranked alternative, this difference being highly significant on the random sample of 100 pairs. For the false positive dataset there was no significant difference in the mean of the log frequency (7.4 for the correct element vs. 7.2 of the network's incorrectly predicted element, $t < 1$). Notice also that the overall mean is much higher for the true positives than the false positives. This result can be explained by observing that in the case of the true positives the frequency distribution of the connection paths is more skewed, with the correct alternative having a much higher frequency than the other alternatives. This seems to indicate that it is more difficult for the RNN to express a preference when it cannot draw information directly from the frequency distribution of the alternative connection paths.

Since the network performance is better than the frequency heuristic, we can conclude that the connection path frequency is an important factor that determines accuracy, but that the decision strategy of the network takes other factors into account.

D. Filtering out the connection paths frequency effect

The following set of experiments aims at understanding what information is exploited by the trained RNN in those cases where it makes a correct prediction, but when the preferred connection path is *not* the most frequent one. We isolate these cases (which represent 10% of the correct prediction cases) and we analyze their characteristics as we have

[5]This is because the connection path frequency follows a Zipfian (log log) distribution.

previously done. In Table IV we report the average value of the significant features that discriminate between the correctly predicted element and the most frequent element and the relative difference of the values. We observe how the RNN has preferred slightly more complex alternatives in terms of heights or number of nodes, but has preferred cases characterized by anchors with a smaller outdegree and at a higher distance from the root. This confirms the importance played by frequency and simplicity on the connection path, but indicates a preference for deeper anchors—in other words, a preference for lower attachment points in the tree. We therefore try to decompose the preference task into two sub-tasks (as reported in section II-B and Fig.2): the first one consists in finding the correct attachment point of the connection path, and the second one consists in choosing the correct connection path itself. Given the previous findings, we hypothesize that the network employs a somewhat more complex decision strategy to disambiguate the attachment point, but that it then exploits only the connection path's frequency to choose the appropriate connection path to attach at that point.

Description	correct	most freq.	Δ / sd
tree max outdeg	4.38	4.69	0.16
tree height	7.74	7.45	0.07
# of nodes in tree	31.01	30.80	0.01
cpath height	1.60	1.43	0.24
# nodes in cpath	2.87	2.62	0.25
anchor outdegree	2.49	4.77	1.43
anchor depth	6.58	3.13	0.88

TABLE IV

COMPARISON BETWEEN RNN AND FREQUENCY HEURISTIC, COMPARING MEANS FOR STRUCTURAL CHARACTERISTICS. TABLE SHOWS FEATURES WHERE PAIRWISE COMPARISONS WERE SIGNIFICANT AT $p < .05$.

E. Analysis of attachment preference

We measure the accuracy of the RNN in determining the correct attachment position along the right frontier. We proceed by grouping all the incremental trees that share the same attachment node. We then rank the groups according to the highest score given by the RNN to any member of the group. We consider a group correctly predicted iff the correct connection path belongs to the group that is ranked highest. The prediction accuracy achieved is 91.5%. The baseline in this case (a random predictor that chooses randomly a connection path) has an accuracy of 37.6%, that is, how many connection paths are in the correct group in respect to the total number of connection paths in all groups averaged over all the forests. Remarkably, if we consider how many times the RNN correctly predicts the anchor attachment within the three best ranked alternatives we achieve an accuracy of 98.4%.

F. Analysis of connection path preference

In this section we study the disambiguation of alternative connection paths and we analyze the relation between frequency-based predictions and RNN predictions.

In the first experiment, we assume that an oracle is available that chooses the correct anchor. In this setting, the network ranks the correct connection path in first position with a 89.5% accuracy. Since the predictions made by the RNN and the most frequent connection path predictor are highly overlapping (91.4%), we refined the analysis considering the true positive (the correct alternative is predicted) and found that in only 3.8% of these cases the preferred incremental tree had a connection path that was not the most frequent; considering the false positives (prediction is not correct) we found that the RNN preferred a more frequent connection path (instead of the correct less frequent one) 66.3% of the time. We conclude that the RNN does exploit frequency information once the correct anchor is given.

This is not necessarily a negative finding and high error rate can also be expected for human first-pass disambiguation decisions, that are also biased by frequencies [15]. We remark that there are no experimental results concerning human performance on first-pass disambiguation carried out on large corpora. Wrong first-pass decisions probably do not have a dramatic impact on the overall performance of the human parser, thanks to its ability to revert to alternative, non-preferred analysis later on in the parsing process. This ability might be realized in terms of a serial backtracking strategy, or alternatively, in terms of the re-ranking of parallel alternatives.

In a second experiment, we assumed that the anchors were predicted by the trained RNN. More precisely, after ranking all the network's prediction we extracted the anchors, removed duplicates after the first occurrence and obtained a rank over the anchors. We then collected all the connection paths that matched one of the first $1, \ldots, i$ anchors and compared the rankings obtained using either the path frequency or the preference of the RNN. In Table V we report the results obtained for $i = 1, 2$ and 3.

i	Frequency	RNN	Relative error reduction
1	88.3	89.5	10.0 %
2	77.35	84.4	31.1 %
3	75.9	83.3	30.7 %

TABLE V

COMPARISON BETWEEN FREQUENCY HEURISTIC AND RNN IN ACCURACY OF CONNECTION PATH DISAMBIGUATION, GIVEN THE ANCHOR PREDICTED BY THE RNN.

These experiments and those reported in Section V-C confirm the initial hypothesis: the RNN is very successful in predicting the anchor and relies mainly on frequency information to predict connection paths. However, the last experiment also shows that the RNN can learn more than pure corpus frequency.

In order to gain a better insight on the kind of statistics that the network is really employing, we assume the working hypothesis that the human parser and the RNN share some common mechanism for ambiguity resolution. We then simulate some known heuristics that have been found in psycholinguistic research, and investigate to what extent they are matched by the network.

G. Comparison to psycholinguistic heuristics

Among the purely structural preferences expressed by the syntactic module of the human parser, psycholinguistic studies identify the minimal attachment (MA) preference and the late closure (LC) preference ([36]). The minimal attachment preference suggests that humans tend to prefer simpler and shorter analysis. In our framework, this translates to preferring connection paths having fewer nodes, which generally implies shorter connection paths. For example, choice 1 would be the preferred one in Fig.2 (b). The late closure preference suggests that humans prefer to connect the current word with more recently processed material. In our framework, this is equivalent to preferring low attachments, i.e. deeper anchors. For example, choice 1 would be preferred in Fig.2 (a). Since a single preference scheme would lead to a number of ties, we first apply one scheme and then break ties by applying the other one. Should ties still occur we resort to the frequency of connection paths. There are two possible combinations: LC-over-MA and MA-over-LC. Results are presented in Fig.6. In order to test whether the RNN has learned to express a

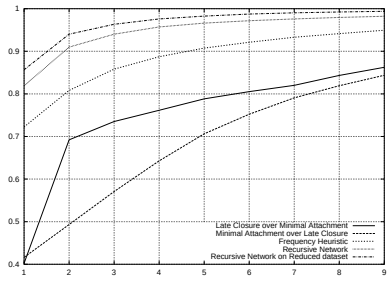


Fig. 6. Comparison between RNN and psycholinguistic heuristics

preference that mimics that of the heuristics, we measure the overlap between pairs of predictors. We report our results in Table VI. In the first row we count how many times two methods rank the same element in first position. In the second row we count how many times there is one common element in the first two ranked positions.

The results indicate that the network choices coincide with those of the heuristics only in roughly half of the cases. If we allow the first or second choice of the network to match either the first or second choice of the heuristic combination we find that the preferences expressed by the RNN and by the LC-over-MA heuristic agree in more than 78% of the cases.

Pos	RNN/LC->MA	RNN/MA->LC	LC-¿MA/MA->LC
1	43.5%	44.5%	91%
1 or 2	78.3%	61.5%	94.4%

TABLE VI

OVERLAPPING PREFERENCES: RNN VS.HEURISTIC AND HEURISTIC VS. HEURISTIC

We can conclude that the network uses a criterion similar to LC-over-MA heuristic combination, but exploits more complex information for those cases in which the heuristic does not apply. This accounts for the 66% error reduction when comparing the prediction accuracy of the RNN to the heuristic combination.

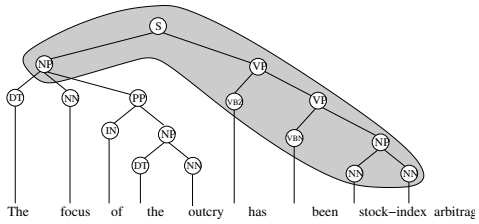


Fig. 7. Tree simplification: the shaded area shows the subset of nodes included in the simplified tree for the attachment of the final word.

VI. ENHANCEMENTS

A. Tree simplification

The experimental results reported in Section V have shown how the complexity of the incremental trees negatively affects the prediction performance. We would like to decrease this complexity (i.e. the number of nodes) without taking the risk of disregarding useful features.

Intuitively not all the information of the incremental tree is significant for the disambiguation task. Specifically, it can be argued that the knowledge of the internal composition of "closed" constituents, i.e. constituents that have been fully parsed, can be summarized by the non-terminal tag that immediately dominates the constituent. In other words we conjecture that the knowledge that a deeply nested NP is made of a sequence of (DT NN) or rather a more complex (DT JJ NN NN) is not much more informative when deciding how to attach a connection path. If this hypothesis is true it should be possible to eliminate a significant part of the nodes of the incremental tree without decreasing the discriminating power of the information that is left in the remaining nodes. We propose a reducing scheme where we keep all the nodes that dominate incomplete components plus all their children. Because of the incremental nature of the algorithm, it turns out that these nodes belong to the right frontier of the incremental tree, or to the children of such nodes. The procedure we are adopting turns out to be consistent with the notion of c-command[6] in theoretical linguistics. When we create T_i, we keep only those nodes that c-command the right frontier of T_{i-1}, plus the right frontier of T_{i-1} itself. Preserving the nodes that c-command the nodes that are active (those that are potential anchors) is linguistically motivated in that it keeps the nodes that can exhibit a "linguistic influence" on each other. In Fig.7 we show the subset of retained nodes.

In order to test the equivalence hypothesis we have run an experiment with the following setting. The datasets are the standard training, validation and test sets where we have applied the simplification procedure. We employ a recursive network having $n = 20$ units and an output network. We report in Fig.6 the comparison between the performance on the reduced dataset and the normal dataset. We observe an increase

[6]A node A c-commands a node B if B is a sister of A or descendent of a sister of A [37].

of performances from 81.7% to 84.82% with a relative error reduction of 17%. The results indicate that the simplification procedure preserves relevant information; in fact, we have helped the system by eliminating potential sources of noise, making the task somewhat simpler and allowing for a better generalization. To explain this behavior we can hypothesize the fact that the states that encode the information relating to deeply embedded nodes (i.e. those that are structurally more distant from the right frontier) are "noisy" and confound less embedded states (i.e. those closer to the frontier).

B. Modular networks

When the learning domain can naturally be decomposed into a set of disjoint sub-domains, it is possible to specialize several learners on each sub-domain. A special case for these specialized learners is when we have informationally encapsulated "modules" [38], that is, predictors whose internal computation is unaffected by the other modules. The linguistic data that we are processing present an intuitive decomposition: the knowledge needed to process the attachment of verb-footed connection paths is quite different from the knowledge used to attach article or punctuation-footed connection paths. It seems plausible that the features that are relevant for discriminating the correct incremental trees are different when dealing with connection paths that have different foots. If there is no significant information overlap between the different cases, we can partition the dataset and select a smaller sub-set of examples with which to train a predictor. The adoption of a modular approach moreover allows a tighter parameter tuning of each module. The knowledge of the domain suggests that certain attachment decisions are harder than others. For example, prepositional phrase attachment is notoriously a hard problem, especially when lexical information is not used (which is our case). In order to determine the "hardness" of each sub-task we setup an experiment that has the following setting. We divide the set of POS tags into $s = 10$ sub-sets where we collate "similar" tags, i.e. tags that have a similar grammatical function[7]. A special set contains all those tags that could not be put in any other sub-set[8]. We employ a network having $n = 25$ units and adopt the same training/validation/test as introduced in Sec.IV-A. The dataset has been pre-processed with the simplification scheme introduced in the previous section. The prediction results are collected and partitioned into the appropriate sub-sets according to which POS tag was involved in the attachment decision. We report the results in Table VII, where column R_u shows the best accuracy obtained using the method of Section VI-A, while column Size reports the fraction of the total dataset represented by each sub-set. The results indicate that the problem is harder in the case of adverbs and prepositions and easier for nouns, verbs and articles. We propose to enhance the overall performance by letting single networks to concentrate on specific ambiguities, i.e. having a RNN being exposed only to attachment decisions involving, for example, adverbs or prepositions. The network specialization can be done in an online or batch fashion. The online scheme is realized using a single network with p different "switching" weight sets for the recursive and output networks. Here the POS tag of the current word selects the appropriate weight set. The batch scheme is realized by pre-processing the training and test sets obtaining p different sub-sets according, once again, to the POS tag and then employing p different networks each one exposed only to uniform attachment decisions. Since the latter solution allows an easier parallelization of the training and testing phase, we resorted to the batch approach. We run two experiments. In the first one we replicated the training set of [39], applied the reduction pre-processing, trained the modular network and tested the performance of the new architecture. We report in column R_{500} of table VII the results. We obtained a total precision in first position of 80.57% against the previous result of 74.0% [39] yielding a 25% relative error reduction. On a second experiment we trained a network of 25 units on the standard training set of 40k sentences and we tested the resulting network on the standard 2k sentences test set. We report the comparison between the performance of the specialized networks (column R_s) and the unspecialized network (column R_u) on the same dataset and the relative error reduction in the last column.

Category	Size %	$R_{500}\%$	R_u %	R_s %	RER %
Adjective	7.48	85.24	87.00	89.46	18.92
Adverb	4.26	45.05	53.46	59.44	12.85
Article	12.45	83.58	89.09	90.99	17.42
Conjunction	2.31	59.55	70.41	78.69	27.98
Noun	32.97	91.84	94.52	95.74	22.26
Other	0.69	51.69	68.64	72.88	13.52
Possessive	2.03	89.75	97.99	97.12	-43.28
Preposition	12.63	61.78	64.26	68.19	11.0
Punctuation	11.72	68.21	75.29	80.84	22.46
Verb	13.46	90.87	94.72	96.77	38.83
Weighted tot	100	80.56	84.82	87.52	17.79

TABLE VII

SPECIALIZATION IMPROVEMENT.PRECISION RESULTS AFTER TRAINING ON THE ORIGINAL 500-SENTENCE TRAINING SET WITH SPECIALIZED NETWORKS (R_{500}), AND AFTER TRAINING ON THE 40K TRAINING SET WITH AN UNSPECIALIZED NETWORK (R_u) AND SPECIALIZED NETWORKS (R_s). RELATIVE ERROR REDUCTION IS SHOWN IN THE FINAL COLUMN.

The results indicate that the specialization procedure results in an overall enhancement of the performance (17.79% relative error reduction in respect of the unspecialized network, 52% relative error reduction in respect to the previous result of 74.0% [39]) and that some categories greatly benefit from this approach[9]. We believe that the reason is that the resources (i.e. areas in the state space) allocated for discriminating the less frequent classes (conjunctions, punctuation, adverbs) do not have to compete against the ones allocated for the most frequent cases (nouns, verbs).

[7]For example all the tags MD VB VBD VBG VBN VBP VBZ , which correspond to modal verbs and verbs with various different tense and agreement information, are grouped together under the category VERB.

[8]It includes POS tags that denote foreign words, exclamations, symbols, etc

[9]Note that the result reported for the possessive case, due to the limited number of examples, does not show any statistically significant difference between the specialized and unspecialized case.

VII. Conclusions

We have shown how the analysis of the preferences expressed by the RNN gives a useful insight into the nature of the statistical information used by the system.

We have found that the RNN bases its preferences to disambiguate the attachment point on complex structural information, but mainly resorts to frequency in choosing the correct connection path. We have moreover shown how the system prefers to attach simple structures to recently processed material, modeling human heuristics, but that the incremental tree offers a richer context on which to condition the preferences.

Taking advantage of the highly structural nature of the domain we have been able to propose a simplification scheme and a specialized architecture that have enhanced the overall prediction accuracy of the network.

We believe that further improvements are achievable introducing more information by lexicalizing the underlying grammar. Future work will focus on the use of the RNN as an informant to guide an incremental parser.

References

[1] G. Altmann and M. Steedman, "Interaction with context during human sentence processing," *Cognition*, vol. 30, pp. 191–238, 1988.

[2] W. Marslen-Wilson, "Linguistic structure and speech shadowing at very short latencies," *Nature*, vol. 244, pp. 522–533, 1973.

[3] M. J. Pickering and M. J. Traxler, "Plausibility and recovery from garden paths: An eye-tracking study," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 24, no. 4, pp. 940–961, 1998.

[4] M. Bader and I. Lasser, "German verb-final clauses and sentence processing," in *Perspectives on Sentence Processing*, C. Clifton, L. Frazier, and K. Rayner, Eds. New Jersey: Lawrence Erlbaum Associates, 1994.

[5] Y. Kamide and D. C. Mitchell, "Incremental pre-head attachment in Japanese parsing," *Language and Cognitive Processes*, vol. 14, pp. 631–632, 1999.

[6] E. P. Stabler, "Parsing for incremental interpretation," 1994, manuscript, University of California at Los Angeles.

[7] B. Roark and M. Johnson, "Efficient probabilistic top-down and left-corner parsing," in *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, 1999, pp. 421–428.

[8] P. C. R. Lane and J. B. Henderson, "Incremental syntactic parsing of natural language corpora with simple syncrony networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 2, 2001.

[9] J. Henderson, "Neural network probability estimation for broad coverage parsing," in *Proceedings of the 10th conference of the European Chapter of the Association for Computational Linguistics (EACL 2003)*, Budapest, Hungary, 2003, pp. 131–138.

[10] A. N. Jain, "Parsing complex sentences with structured connectionist networks," *Neural Computation*, vol. 3, pp. 110–20, 1991.

[11] C. Kemke, "A constructive approach to parsing with neural networks - the hybrid connectionist parsing method," in *Proceedings of 15th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2002*, vol. 2338, Calgary, Canada, 2002, pp. 310–318.

[12] R. Miikkulainen, *Subsymbolic Natural Language Processing: an integrated model of scripts, lexicon and memory*. MIT Press, 1993.

[13] S. Wermter and V. Weber, "SCREEN: learning a flat syntactic and semantic spoken language analysis using artificial neural networks," *Journal of Artificial Intelligence Research*, vol. 6, pp. 35–85, 1997.

[14] F. Cuetos and D. C. Mitchell, "Cross-linguistic differences in parsing: Restrictions on the use of the late closure strategy in Spanish," *Cognition*, vol. 30, pp. 72–105, 1988.

[15] D. C. Mitchell, F. Cuetos, M. M. B. Corley, and M. Brysbaert, "Exposure-based models of human parsing: Evidence for the use of coarse-grained (nonlexical) statistical records," *Journal of psycholinguisitic research*, vol. 24, 1995.

[16] P. Sturt, F. Costa, V. Lombardo, and P. Frasconi, "Learning first-pass structural attachment preferences using dynamic grammars and recursive neural networks," *Cognition*, vol. 88, pp. 133–169, 2003.

[17] V. Lombardo and P. Sturt, "Incrementality and lexicalism: A treebank study," in *Lexical Representations in Sentence Processing*, S. Stevenson and P. Merlo, Eds. John Benjamins, 1999.

[18] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Transactions on Neural Networks.*, vol. 9, pp. 768–786, 1998.

[19] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, 1997.

[20] M. Collins and N. Duffy, "Convolution kernels for natural language," in *Proc. of NIPS*, 2001.

[21] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: the Penn Treebank," *Computational Linguistics*, vol. 19, pp. 313–330, 1993.

[22] S. Stevenson, "Competition and recency in a hybrid network model of syntactic disambiguation," *Journal of Psycholinguistic Research*, vol. 23, no. 4, pp. 295–321, 1994.

[23] T. Vosse and G. Kempen, "Syntactic structure assembly in human parsing: a computational model based on competitive inhibition and a lexicalist grammar," *Cognition*, vol. 75, pp. 105–143, 2000.

[24] E. Brill, "A simple rule-based part-of-speech tagger," in *Proceedings of ANLP-92, 3rd Conference on Applied Natural Language Processing*, Trento, IT, 1992, pp. 152–155. [Online]. Available: citeseer.nj.nec.com/brill92simple.html

[25] D. Milward, "Dynamic dependency grammar," *Linguistics and Philosophy*, vol. 17, no. 6, 1994.

[26] V. Lombardo and P. Sturt, "Towards a dynamic version of tag," in *Proceedings of the TAG+6 Workshop*, 2002.

[27] ——, "Incremental processing and infinite local ambiguity," in *Proceedings of the 19th Annual Conference of the Cognitive Science Society, Stanford CA*, 1997, pp. 448–453.

[28] H. Thompson, M. Dixon, and J. Lamping, "Compose-reduce parsing," in *Proceedings of the 29th Meeting of the Association for Computational Linguistics*, Berkley, California, June 1991, pp. 87–97.

[29] M. Collins, "Three generative, lexicalised models for statistical parsing," in *Proceedings of the 35th annual meeting of the Association for Computational Linguistics*, 1997, pp. 16–23.

[30] W. W. Cohen, R. E. Schapire, and Y. Singer, "Learning to order things," in *Advances in Neural Information Processing Systems*, M. I. Jordan, M. J. Kearns, and S. A. Solla, Eds., vol. 10. The MIT Press, 1998.

[31] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf, "Ranking on Data Manifolds," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA, USA: MIT Press, 2004.

[32] C. Goller and A. Kuechler, "Learning task-dependent distributed structure-representations by back-propagation through structure," in *IEEE International Conference on Neural networks*, 1996, pp. 347–352.

[33] M. J. Collins, "A new statistical parser based on bigram lexical dependencies," in *Proceedings of the 34th annual meeting of the Association for Computational Linguistics*, 1996.

[34] E. Charniak, "Expected-frequency interpolation," Technical report, CS96-37, Department of Computer Science, Brown University, 1996.

[35] J. Heaps, *Information Retrieval–Computational and Theoretical Aspects*. New York, NY: Academic Press, Inc., 1978.

[36] L. Frazier, "On comprehending sentences: Syntactic parsing strategies," Ph.D. dissertation, University of Connecticut, Storrs, CT, 1978.

[37] N. Chomsky, *Lectures on Government and Binding*. Foris, 1981.

[38] A. Sharkey, "On combining artificial neural nets," 1996.

[39] F. Costa, P. Frasconi, V.Lombardo, and G. Soda, "Towards incremental parsing of natural language using recursive neural networks," *Applied Intelligence*, vol. 19, no. 1–2, pp. 9–25, 2003.

Fabrizio Costa received the Laurea degree in Electronic Engineering in 1998, and the Ph.D. degree in Computer Science in 2002, both from the University of Florence, Italy. His research interests include machine learning in structred domains with connectionist and kernel based approaches. In his reserches he has been working in the natural language processing and bioinformatic domain.

Paolo Frasconi received the M.Sc. degree in Electronic Engineering in 1990, and the Ph.D. degree in Computer Science in 1994, both from the University of Florence, Italy, where he is presently an Associate Professor of Computer Science. He previously held positions at the University of Cagliari, Italy, at the University of Wollongong, Australia, and the Massachusetts Institute of Technology. His current research interests are in the area of machine learning using connectionist models and belief networks, with particular emphasis on problems involving learning about sequential and structured information. Application fields of his interest include bioinformatics, natural language processing, pattern recognition, and document processing. Dr. Frasconi serves as an Associate Editor for the IEEE Transactions on Neural Networks, the IEEE Transactions on Knowledge and Data Engineering, and the ACM Transactions on Internet Technology. In 2001 he co-directed the NATO Advanced Studies Institute "Artificial Intelligence and Heuristic Methods for Bioinformatics." He is a member of the ACM, the IAPR, the IEEE, and the AI*IA.

Vincenzo Lombardo , born 1964, is an Associate Professor at the Department of Computer Science and the School of Multimedia and Arts of the University of Turin. He received a Laurea degree in Computer Science from the University of Turin in 1987 and PhD in Computer Science from the conjoined Turin-Milan PhD Programme in 1993. His current research areas are Natural Language Processing (syntax, parsing, computational psycholinguistics), AI for art and drama, computer music.

Patrick Sturt received the MA degree in linguistics from University College London, and the PhD degree in Cognitive Science from the University of Edinburgh. He is currently a Senior Lecturer in Psychology at the University of Glasgow. His research interests include the experimental study of human sentence processing using eye-movement recording techniques, and computational models of parsing.

Giovanni Soda received his degree in Mathematics in 1969 from the University of Florence (Italy). From 1971 to 1982 he was a researcher at the National Council of Research (CNR) where his activity included formal systems for language manipulation, and data structures. From 1983 to 1991 he was Associate Professor of Programming Languages at the Faculty of Electronic Engineering of the University of Florence, where he is presently Full Professor of Artificial Intelligence. His current interests include Knowledge Representation Systems, integration of Artificial Intelligence techniques with Databases, Neural Networks and document processing. He is author of about one hundred papers in international journals, contributed volumes, and conference proceedings on several aspects of computer science and artificial intelligence. Professor Soda is a member of IEEE, ACM and a member of the Steering Committee of the AI*IA (Associazione Italiana per l'Intelligenza Artificiale). He is also Editor-in-Chief of the AI*IA Notizie.