# Towards incremental parsing of natural language using recursive neural networks

F. Costa[1], P. Frasconi[1], V. Lombardo[2], and G. Soda[1]
[1]Dept. of Systems and Computer Science, University of Florence
via di Santa Marta 3, I-50139 Firenze (Italy)
[2]DISTA, Università del Piemonte Orientale "A. Avogadro"
c.so Borsalino 54, I-15100 Alessandria (Italy)

January 18, 2001

**Abstract**

In this paper we develop novel algorithmic ideas for building a natural language parser grounded upon the hypothesis of incrementality. Although widely accepted and experimentally supported under a cognitive perspective as a model of the human parser, the incrementality assumption has never been exploited for building automatic parsers of unconstrained real texts. The essentials of the hypothesis are that words are processed in a left-to-right fashion, and the syntactic structure is kept totally connected at each step.

Our proposal relies on a machine learning technique for predicting the correctness of partial syntactic structures that are built during the parsing process. A *recursive* neural network architecture is employed for computing predictions after a training phase on examples drawn from a corpus of parsed sentences, the Penn Treebank. Our results indicate the viability of the approach and lay out the premises for a novel generation of algorithms for natural language processing which more closely model human parsing. These algorithms may prove very useful in the development of efficient parsers.

## 1 Introduction

Incremental processing of the human parser is a widely held assumption in psycholinguistics. This assumption accounts for the intuitive fact that language is processed from left to right: this is obviously true for speech, which is obligatorily fed to the processor in a

temporal sequence; it is less obvious for text, where the bidimensional language transcription (as opposed to the monodimensional speech) allows an inspection of linguistic material that goes beyond the left-to-right sequential presentation. Incremental processing also accounts for a fact which has received much experimental support, i.e. that humans interpret language without reaching the end of the input sentence, that is they are able to assign a meaning to "almost any" initial (left) fragment of a sentence (Marslen-Wilson, 1973).

Although incremental processing of natural language is uncontroversially supported by many psycholinguistic experiments, effective models for parsing unconstrained language are rarely based on the incrementality hypothesis. In this area, most solutions consist in stochastic versions of bottom-up parsers, relying on models grounded upon context-free rule probabilities (see, e.g. (Collins, 1996)). The incremental approach is at work only in psycholinguistic models of the human parser, mostly tested on artificially-generated sentences that display this or that ambiguity form. Probably, the major reason is the incomplete knowledge about the human parser, which makes a reliable algorithmic design considerably hard.

Nevertheless, some of the discovered features of the human parser have been exploited in NLP system implementations. For example, a number of preference heuristics (like Late Closure (Kimball, 1973) or Minimal Attachment (Frazier & Fodor, 1978), see Section 5), devised in psycholinguistic environments to describe some aspects of the human parser, have been implemented to solve local ambiguities (see, e.g., (Hobbs & Bear, 1990)). After the emergence of the corpus-based approaches, NLP systems often involve a combination of heuristics and statistics to guide the parser decisions (see, e.g., (Nagao, 1994)). In this paper we push further the similarities with the human parser, by adopting the novel assumption that the parsing strategy incorporates an architectural aspect of the human parser, namely its incremental nature. To our knowledge, this is the first attempt in which a psycholinguistic model of the human parser is applied to parsing real texts. The sense of "towards" in the paper title is that the result presented here is not a fully functional system yet, but a concrete assessment of the proposed methodology.

An operational account of the incrementality hypothesis, called *strong incrementality*, is at the core of a number of computational models of the human parser, as a parsimonious algorithmic version of incrementality (Stabler, 1994). In this version, the parser maintains a totally connected parse, while scanning the input words from left to right, with no input stored in a disconnected state. The major difficulty of implementing a strongly incremental

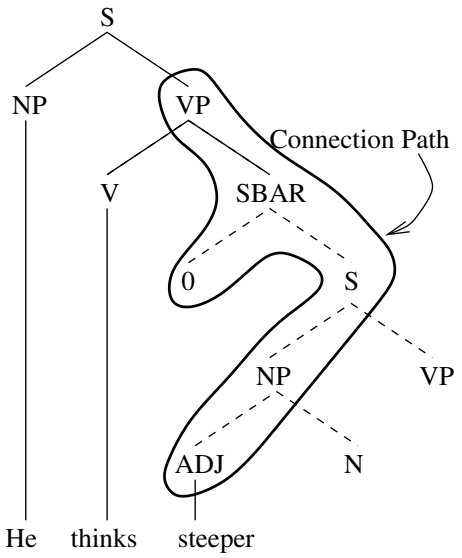



Figure 1: The connection path for "steeper" in "He thinks steeper prices are to come."

strategy is that the connection of the next input word to the existing structure (what is called the *left context*) requires the construction of a substructure called the *connection path* (see Figure 1). In parsing naturally occurring text, this results in a high number of candidate connection paths, which yields a hard search problem, affecting the parser accuracy and speed.

In this paper we propose an incremental parsing model able to process unconstrained naturally occurring texts. At each step of parsing, the correct connection path linking the next word to the incrementally built parse tree is predicted using a neural network model. Such a prediction effectively acts like a heuristic that can guide the search process, and significantly reduce computational efforts. Designing accurate heuristics for this problem is a difficult task but, on the other hand, the availability of databases of parsed sentences (treebanks) makes the problem interesting from a machine learning perspective. In our experiments, examples are drawn from a subset of parsed sentences in the Penn Treebank (Marcus et al., 1993).

As far as we know, there exists one previous attempt of applying machine learning techniques to formulate parse decisions. Hermjakob and Mooney (Hermjakob & Mooney, 1997) used a modified ID3 algorithm to learn the moves of a shift-reduce parser. The problem is cast into a representation which relies on a rich set of morphological, syntactic and semantic features (around 200). The result is a parsing model that provides an output consisting

of a phrase structure tree and a predicate-argument structure fed to a machine translation module. Though similar in the spirit, the work described in this paper relies on a different parsing model, a strongly incremental parser vs. a shift-reduce parser, and a different machine learning technique, a recursive neural network vs. ID3 algorithm. Also, the goal of this paper is the assessment of a methodology rather than a fully functional system: the numerical data we provide result from a reduced learning domain which is exclusively syntactic (vs. the 200 features from several sources), in the form of syntactic structures. We are aware that a complete parsing model cannot be realized without the contribution of other knowledge sources, like a semantic knowledge base and a set of subcategorization frames (or predicate-argument structures), but the results we have yielded with the skeletal framework clearly indicate the effectiveness of the approach.

Instances in our learning domain are naturally represented as incremental trees, obtained by joining a candidate connection path to some left context (another incremental tree). Uncertainty about the selection of the correct candidate can be naturally modeled in a probabilistic setting. In particular, the prediction task consists of estimating the probability that a given attachment between a given incremental tree and a candidate path is correct. As a major advantage of this formulation, the learner outputs a continuous ranking of alternatives so that best candidates can be tried first.

The probabilistic and supervised nature of the learning task (correct candidates are known for parsed sentences in the training set) suggests a solution based on connectionist models. The structured nature of instances requires the adoption of a recently introduced class of connectionist architectures, called *recursive neural networks* (Goller & Küchler, 1996; Sperduti & Starita, 1997; Frasconi et al., 1998). Unlike feedforward neural networks, which are grounded on attribute-value representations of data, recursive networks can adaptively process labeled graphs, effectively taking advantage of relations among atomic entities that constitute a single instance. Representation and processing of structured information is not a recent area of interest in connectionism. Early attempts include distributed reduced descriptors by Hinton (1990), Recursive Auto-Associative Memories (RAAM) by Pollack (1990) and Holographic Reduced Representations by Plate (Plate, 1995). Recursive neural networks, however, are the first architecture that can exploit the supervised learning paradigm on structured data. They have been successfully applied to chemistry (Bianucci et al., 2000), pattern recognition (Francesconi et al., 1997), and learning search heuristics for guiding a theorem prover (Goller, 1997). In this paper we describe a novel and specialized version of these networks that enables them to learn how to rank a forest of

alternatives.

The paper is organized as follows: Section 2 introduces the basic notions of incremental natural language processing. In section 3 we formulate the learning task. Section 4 reviews the basic neural network architecture presented in (Frasconi et al., 1998) and specialize it to solve the learning problem of the present paper. In section 5 we explain in detail data preparation and the experimental setup, with a discussion of the results obtained so far.

## 2 Incremental parsing

Incremental processing of natural language (incrementality for short) is an intuitively plausible hypothesis upon the human language processor.

This incrementality hypothesis has received a large experimental support in the psycholinguistic community over the years: the shadowing experiments (Marslen-Wilson, 1973) proved that people are able to interpret speech word-by-word with a latency time of just 250 ms[1]; data about head-final language processing[2] suggest that the human parser assembles substructures eagerly before reaching the structure head (Bader & Lasser, 1994; Frazier, 1987; Kamide & Mitchell, 1999; Yamashita, 1994); finally, eye-movement studies of visual object recognition show that the referent in a visual context of a noun phrase with pre-modifiers is computed well before reaching the nominal head (Eberhard et al., 1995).

The incrementality hypothesis poses some constraints upon the general parsing strategy. In its general form, the hypothesis implies that the semantic interpretation of some fragment of the sentence is available as the scan of the input material proceeds from left to right. This does not involve any specific constraint on how the interaction between the syntactic analysis and the semantic interpretation can be implemented at the algorithmic level. One possible (and parsimonious) solution to the interaction problem is offered by the strongly incremental version of the hypothesis (Lombardo et al., 1998; Milward., 1995; Stabler, 1994; Steedman, 1989; Sturt & Crocker, 1996). The interaction occurs at the word level, and

[1] In a shadowing experiment, subjects listen to passages from a novel and say words out loud as fast as they can. Subjects do not know that certain words in the passage are mispronounced. In many cases, subjects tend to utter words correctly with no loss of time with respect to the correct pronunciation. These and other data have been interpreted as a proof that people tend to interpret the meaning of sentences as soon as they can, probably on a word-by-word basis.

[2] Head-final languages are languages where constituents feature the head word in the rightmost position. Examples are Dutch, German and Japanese.

compels the syntactic analyzer to keep a totally connected structure at all times (that is, the semantic interpreter has no means to work on disconnected items[3]). Parsing proceeds from left to right through a sequence of incremental trees, each spanning one more word to the right. Here are the formal definitions[4].

Given a sentence $s = w_0 w_1 \cdots w_i \cdots w_{|s|-1}$ and a tree $T$ for it (whose internal nodes are labeled by nonterminal symbols and whose leaves are labeled by words), we define recursively the *incremental trees* $T_i (i = 0, 1, ..., |s| - 1)$ spanning $w_0 \cdots w_i$ as follows:

- $T_0$ consists of the chain of nodes and edges from $w_0$ to its maximal projection;
- $T_i$ consists of all the nodes and edges in $T_{i-1}$ and the chain of nodes and edges from $w_i$ to $N$, where $N$ is
  - either a node of $T_{i-1}$,
  - or the lowest node of $T$ dominating both the root of $T_{i-1}$ and $w_i$ (in this case $T_i$ also includes the edge from N to the root of $T_{i-1}$).

Given two incremental trees $T_1$ and $T_2$, we define the *difference* between $T_1$ and $T_2$ as the tree formed by all the edges which are in $T_1$ and not in $T_2$, and all the nodes touched by such edges.

Now, given a sentence $s = w_0 w_1 \cdots w_{|s|-1}$ and a tree $T$ for it, the *connection path* for $w_i$ is the difference between the incremental trees $T_i$ and $T_{i-1}$. Moreover,

- A node both in $T_i$ and in $T_{i-1}$ is called an *anchor* (that is, a node where the connection path anchors to $T_{i-1}$).
- The node labeled by the POS tag of $w_i$ is called a *foot*.

In Figure 2, we show the sequence of incremental trees for a sentence of the corpus.

[3]This is an assumption upon the semantic interpreter: it could well be that the human processor has some means of interpreting the disconnected pieces of structure in memory. However, the latter is an unparsimonious assumption.

[4]These definitions introduce two simplifications with respect to (Lombardo & Sturt, 1999): first, we include in the connection path the edges that link a word and its maximal projection, that is the substructure headed by that word; second, we neglect the presence of empty nodes (traces) between the lexical elements $w_{i-1}$ and $w_i$. Both simplifications make the design of the learning domain easier, because we are not considering the distinction between headed and headless nodes in connection paths (see Section 5) and co-referring symbols, respectively.

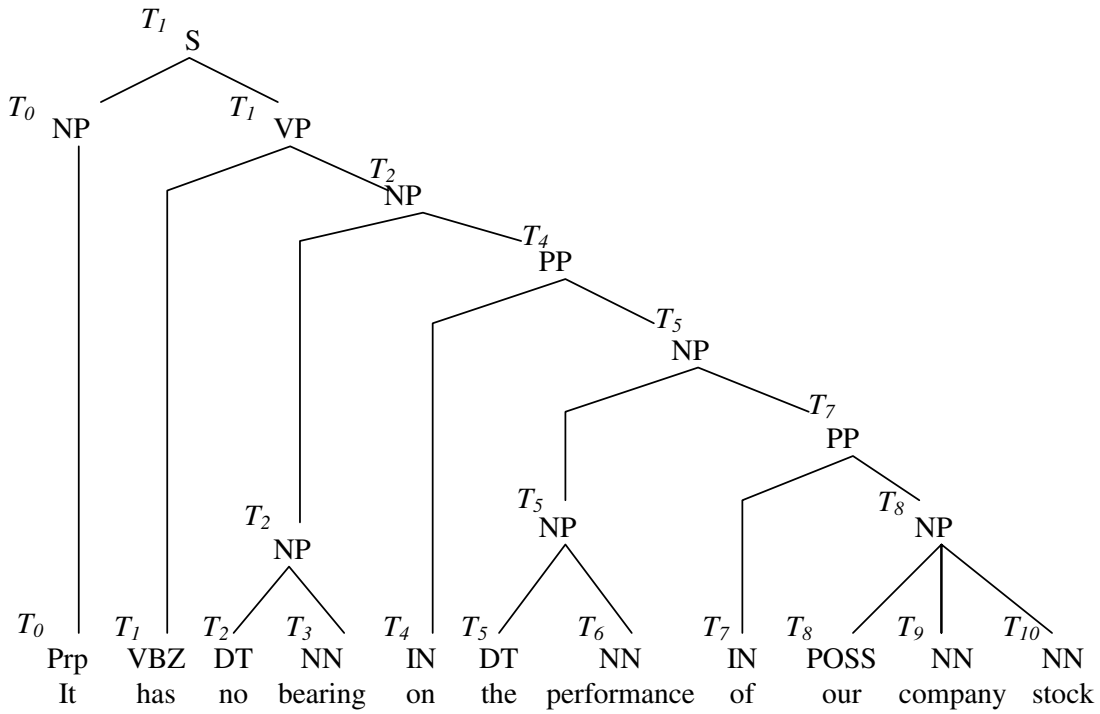
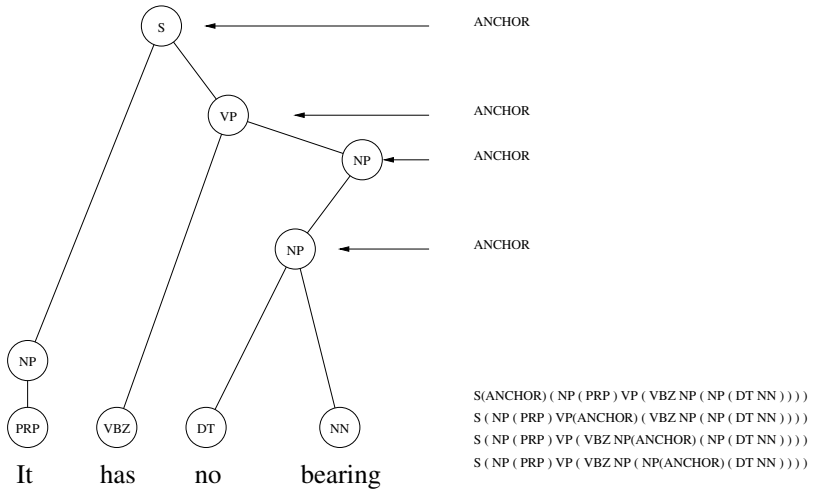
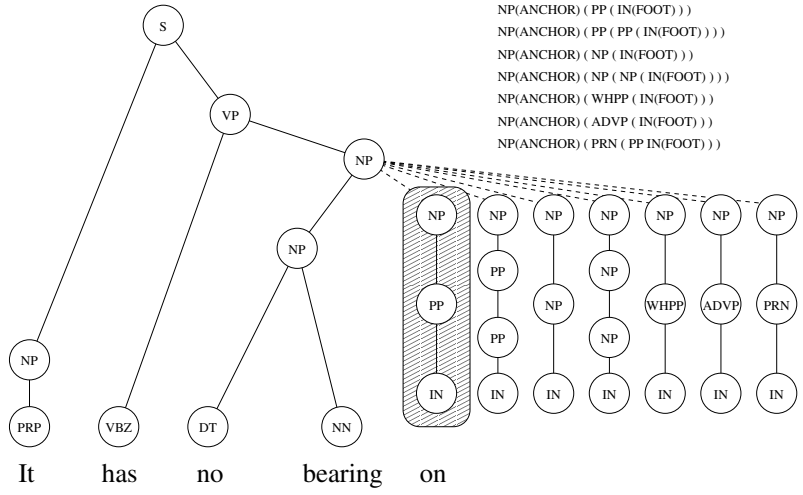



Figure 2: The incremental trees of the sentence "It has no bearing on the performance of our company stock." Nodes are labeled with the incremental tree that includes them for the first time.




(a)




(b)

Figure 3: Legal attachment of a connection path to incremental trees. The sentence in this example is "It has no bearing on the performance of our company stock." (a) An incremental tree $T_{i-1}$; nodes marked as "anchor" are potential anchor points for joining a connection path. (b) After one of the potential anchor points has been selected (NP in the example), several alternative paths can be joined to form $T_i$; in the example, these are all the paths having NP as an anchor and IN (the POS tag of "on") as foot. The correct path for this sentence is highlighted.

When the parser tries to connect an input word $w_i$ to the incremental tree $T_{i-1}$ (i.e. the structure that spans all the words that precede the current word), it computes a connection path, that is the structural portion that must be built in order to yield $T_i$. However, the number of paths that allow to link $w_i$ with $T_{i-1}$ can be high (see Section 5 for a quantitative estimation on the corpus). Ambiguity comes in two forms: the number of possible anchors on the right edge of $T_{i-1}$, and the number of different paths that can link $w_i$ with $T_{i-1}$ from some anchor (see Figure 3). A selection procedure chooses the best connection path and anchor for continuation, and instantiates it to generate the new incremental tree $T_i$. The better is the selection procedure, the less parsing decisions are wrong.

The success of a selection depends on the ultimate task of the parsing model. If the goal is to simulate the human parser (this is interesting from a psycholinguistic standpoint), then the selection procedure must return the connection path that would be chosen by a human, even if it is wrong. In fact, it can happen that first pass analyses lead to a parsing breakdown, and the control goes to a reanalysis module, a quasi-necessary component of a psycholinguistic model (Fodor & Ferreira, 1998). On the contrary, if the goal is the design of an efficient automatic parser, then the selection procedure must minimize the number of failures due to the wrong choice of the connection path. The goal of this paper is to evaluate whether machine learning algorithms can effectively solve the local ambiguity due to the selection of the connection path. Thus, the reference task considered in this paper is the design of an efficient automatic parser. In the next section we formalize the learning task.

# 3 Formal definition of the learning problem

In the following, we assume that a treebank (a corpus of sentences with associated parse trees) is available. A treebank is denoted as $\boldsymbol{B} = \{(s^{(p)}, T(s^{(p)})), p = 1, \cdots, P\}$ where $s^{(p)}$ is a generic sentence and $T(s^{(p)})$ its parse tree.

## 3.1 Universe of connection paths

The *universe of connection paths* $U(\boldsymbol{B})$ is the set of all connection paths that can be extracted from $T(s^{(p)}), p = 1, \cdots, P$. This set of paths can be obtained by running a simulation algorithm that computes all the incremental trees associated with the sentences in the corpus. The algorithm scans the sentence covered by this tree in a word-by-word

fashion from left to right, and for each word $w_i$, finds the legal connection paths which would be built in order to attach $w_i$ to the current left context $T_{i-1}$. The algorithm simulates the building of structure by *marking*, during the scan of the tree, the branches which would be built by a perfectly informed incremental parser. More details are explained in (Lombardo & Sturt, 1999)).

The universe $U(\boldsymbol{B})$ effectively plays the role of a grammar for the sentences in $\boldsymbol{B}$. For a large enough corpus, we can expect that $U(\boldsymbol{B})$ will provide an approximately complete coverage of rules for covering all the sentences in the language under consideration.

### 3.2 The forests of candidates

Suppose we are given a new sentence $s = w_0, \ldots, w_{|s|-1}$ (not in the corpus $\boldsymbol{B}$), and suppose that at stage $i$ of parsing we know the correct incremental tree $T_{i-1}(s)$ spanning $w_0, \ldots, w_{i-1}$. The goal of an incremental parser is then to compute the next tree $T_i(s)$ in order to accommodate the next word $w_i$. Assuming that $U(\boldsymbol{B})$ has a rich enough coverage, we know that $T_i(s)$ can be obtained by joining $T_{i-1}(s)$ to one of the paths in $U(\boldsymbol{B})$. However, other trees spanning $w_1, \cdots, w_i$ can be generated by legally attaching other connection paths. In this regard, recall from Section 2 that an attachment is legal if the anchor matches one of the nodes in the right edge of the tree and the foot matches the POS tag of $w_i$ (see Figure 3). Legal foots can be easily obtained by running a part-of-speech tagger on the sentence. The set of trees obtained by legally attaching $T_{i-1}(s)$ to any path in $U(\boldsymbol{B})$ is called the *forest of candidates* for word $w_i$ within sentence $s$, denoted $\boldsymbol{F}_i(s) = \{T_{i,1}(s), \ldots, T_{i,m_i}(s)\}$, where $m_i = |\boldsymbol{F}_i|$. It is worth noting that distinct sentences having the same associated sequence of POS tags are indistinguishable in our approach.

Of course, only one of the trees in $\boldsymbol{F}_i(s)$, $T_{i,j^\star}(s)$, is the correct one (i.e., it is a subgraph of the actual parse tree $T$ for the sentence) but for unparsed sentences it is unknown. If an oracle were available to select the correct incremental tree from the candidate set, then incremental parsing would become a trivial procedure. The machine learning algorithms in this paper aim to accurately approximate such an oracle.

### 3.3 The learning task

We propose that the learning algorithm rely on a statistical model in charge of assigning probabilities of correctness to each candidate tree. Parameters of the model will be esti-

mated from examples. The main advantage of a probabilistic formulation is that during recall, when unseen sentences are processed, the model can be effectively employed to rank alternative trees, sorting them by increasing probability of correctness. In this way, the parser can try first candidate trees with higher probability of correctness. If we look at parsing as a search problem (where the goal is the full parse tree for $s$) then ranking candidates can be seen as a heuristic evaluation function that guides the search problem. A perfect predictor (i.e., one that always assigns the highest probability to the correct tree) would give a perfect heuristic, reducing the branching factor of the search problem to 1. Performance of the learning algorithm should be evaluated having this aspect in mind. This means that two learners should not be compared by simply computing the number of failures to assign the highest probability to the correct tree (as in a winner-take-all scheme). In facts, we need that the number of candidates ranked above the correct tree is small. Hence, one performance measure for the predictor can be the average position of the correct tree in the list of candidates, after trees have been sorted by increasing probability of correctness.

In this paper, we assume that the model does not make use of lexical information, that is, correctness is assessed by only observing nonterminal symbols in each tree, discarding all the leaves (which are labeled by words). There are two reasons for this choice. First, incorporating lexical knowledge in a connectionist model necessarily involves a large number of parameters to be estimated, requiring large amount of training data. In our experiments (see Section 5) we show that even a limited amount of training sentences is sufficient to achieve interesting performance. Second, we are interested in showing that the pure syntactic level contains significant information under the incrementality assumption. In the framework we are outlining, lexical knowledge can be employed after learning to detect parsing breakdowns and to implement recovery mechanisms.

There are two possible formulations of the learning task:

- Each instance is a candidate incremental tree. The instance space is then partitioned into *positive* and *negative* instances. Candidate $T_{i,j}(s)$ is a positive instance if and only if it is the correct incremental tree for $s$ at position $i$. In this formulation, the learner must classify labeled trees into one of two classes.

- Each instance is the whole forest of candidate trees $\boldsymbol{F}_i(s)$. The task consists of learning the correct member of the forest, which can be identified by a multinomial variable with realizations in $\{1, \ldots, m_i(s)\}$. Training examples are pairs $(\boldsymbol{F}_i(s), j_i^\star(s))$, where

the input portion is the candidate forest and the output portion (supervision) is the integer $j_i^\star(s) \in [1, m_i(s)]$ identifying the correct tree.

While the first formulation relies on the standard concept learning framework, the second formulation requires more attention because it does not involve classification as commonly intended. In fact, one instance is a "bag" of objects (trees), in a way that loosely resembles multi-instance learning (Auer, 1997; Dietterich et al., 1997). In our case (as a major difference with respect to multi-instance learning), each bag contains exactly one positive tree, and such tree can be always identified in the training set.

The importance of our second formulation stems from the fact that it more accurately describes the kind of uncertainty we are dealing with. To explain this, let us consider the probabilistic interpretations of the two formulations. In the first case, the learner makes decisions about boolean output variables $O_{i,j}(s)$, each indicating whether a given candidate incremental tree is correct for position $i$ of sentence $s$. In this case, the output of the learner for word $i$ and candidate $j$, denoted $y_{i,j}(s)$, is interpreted as

$$y_{i,j}(s) = P(O_{i,j}(s) = \text{correct} | T_{i,j}(s)). \tag{1}$$

When computing this probability the learner is not informed about competing candidate trees. In the second formulation, the learner makes decisions about a *multinomial* output variables $O_i(s)$ whose realizations are integers in $[1, \cdots, m_i(s)]$ identifying the correct tree in the bag associated with word $w_i$ in sentence $s$:

$$y_{i,j}(s) = P(O_i(s) = j | \boldsymbol{F}_i(s)) \tag{2}$$

where $O_i(s) = j$ means that $T_{i,j}(s)$ is the correct tree. In this case, predictions are conditional to the whole set of candidates, thus introducing competition among trees in the forest. Assuming that the universe of paths can cover the sentence $s$ (i.e. the grammar is sufficiently expressive), we have

$$\sum_{i=1}^{m_i(s)} y_{i,j}(s) = 1. \tag{3}$$

This is because one and only one candidate tree is the correct one (but which one is unknown). If the grammar is not sufficiently expressive, predictions are meaningless but, on the other hand, in that case parsing would fail anyway. Hence, as far as the learning task is concerned, we can assume that (3) holds.

In the following section we shall explain how this learning task can be solved using a

connectionist approach. To simplify notation, reference to the particular sentence $s$ will be omitted in the following discussion.

# 4 Neural network architecture

Neural network architectures for supervised learning are well known in the case of attribute-value representations. However, as explained in Section 3, the learning task that can instruct parse decisions involves making predictions about structured objects (namely, labeled trees). Feedforward neural networks cannot take as input labeled trees because they cannot deal with structured objects that have variable size, and embed relations among atomic constituents. In principle, recurrent neural networks (Kolen & Kremer, 2000) might be employed by converting the tree into a sequence (e.g., by visiting it in preorder). However, such an approach would lead to difficult learning tasks for at least two important reasons. First, the exponential relation between number of nodes and height in a tree leads to very long sequences even for trees of modest height. This has a very negative impact on learning because of vanishing gradient problems (Bengio et al., 1994). Second, once a tree is linearized into a sequence, hierarchical relations are blurred and intermixed, making generalization to new instances hard (Frasconi et al., 1998). These considerations motivate a different class of architectures, called *recursive* neural networks, that generalize recurrent neural networks to deal with labeled directed acyclic graphs (Goller & Küchler, 1996; Sperduti & Starita, 1997; Frasconi et al., 1998). In Section 4.1 we briefly review the essential architectural and algorithmic ideas for supervised classification of data structures — more details can be found in (Frasconi et al., 1998). In Section 4.2 we reformulate the basic classification model for ranking alternative incremental trees according to the second formulation outlined in Section 3.

## 4.1 Recursive neural networks

From a connectionist (or statistical) standpoint, supervised learning for classification is formalized as the problem of estimating the conditional probability $P(O|I)$. The output $O$ is a discrete variable associated with the class and $I$ is a generic input object. The general theory developed in (Frasconi et al., 1998) allows $I$ to be a directed acyclic graph with a supersource. Here we are interested in the case of the input $I$ being a labeled ordered $m$-ary tree. By ordered we mean that, for each vertex $v$, a total order is defined on the $m$

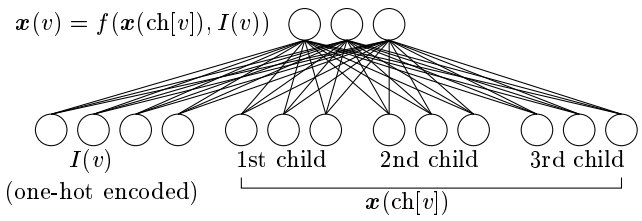



Figure 4: The state transition function $f$ of a RNN is realized by a feedforward neural network.

children of $v$. Specifically, $\mathrm{ch}[v]$ denotes the ordered $m$-tuple of vertices whose elements are $v$'s children. If $v$ has $k < m$ children, then the last $m - k$ elements of $\mathrm{ch}[v]$ are filled with a special entry *nil*, denoting a missing child. $I(v)$ denotes the label attached to vertex $v$. Although the general theory allows to mix discrete, continuous, and possibly multivariate labels, in this paper we are only interested in the case of discrete labels, i.e. random variables with realizations in a finite alphabet $\mathcal{I} = NT_1, \ldots, NT_N$ (in our application, this is the set of nonterminal symbols in the treebank).

The basic neural network architecture is based on the following recursive state space representation:

$$\boldsymbol{x}(v) = f(\boldsymbol{x}(\mathrm{ch}[v]), I(v)) \tag{4}$$

In the above equation, $\boldsymbol{x}(v) \in I\!R^n$ denotes the state vector associated with node $v$ and $\boldsymbol{x}(\mathrm{ch}[v]) \in I\!R^{m \cdot n}$ is a vector obtained by concatenating the components of the state vectors contained in $v$'s children. This vector can be interpreted as a distributed representation of the subtree dominated by vertex $v$. $f : \mathcal{I} \times I\!R^{m \cdot n} \to I\!R^n$ is the state transition function that maps states at $v$'s children and the label at $v$ into the state vector at $v$. States in Eq. (4) are updated bottom-up, traversing the tree in post-order. If a child is missing, the corresponding entries in $\boldsymbol{x}(\mathrm{ch}[v])$ are filled with the *frontier* state $\overline{\boldsymbol{x}}$, which is associated with the base step of recursion. A typical choice is $\overline{\boldsymbol{x}} = 0$. This computational scheme closely resembles the one used by frontier-to-root tree automata (Thacher, 1973), but in that case states are discrete whereas here states are real vectors. After recursion (4) is completed, a distributed representation of the whole tree is obtained in correspondence of the root $r$. Output predictions are then computed as

$$o = g(\boldsymbol{x}(r)). \tag{5}$$

where $g : I\!R^n \to [0, 1]^m$ is the output function, that maps the state $\boldsymbol{x}(r)$ (at the root $r$ of

the tree) into the probability $P(O|I)$ that the input tree is a positive example.

The transition function $f$ is implemented by a feedforward neural network, according to the following scheme (see also Figure 4):

$$a_h(v) = \omega_{h,0} + \sum_{j=1}^{N} \omega_{hj} z_j(I(v)) + \sum_{k=1}^{m} \sum_{\ell=1}^{n} w_{hk\ell} x_\ell(\mathrm{ch}_k[v]) \tag{6}$$

$$x_h(v) = \tanh(a_h(v)) \qquad h = 1, \ldots, n \tag{7}$$

where $x_h(v)$ denotes the $h$-th component of the state vector at vertex $v$, $z_j(NT_q) = 1$ if $j = q$ or zero otherwise (i.e., we are using a one-hot encoding of symbols), $\mathrm{ch}_k[v]$ denotes the $k$-th child of $v$, and $\omega_{ij}, w_{hk\ell}$ are adjustable weights. It is important to remark that weights in the transition network are independent of the node $v$ at which the transition function is applied. This can be seen as a form of weight sharing.

The output function $g$ is implemented by another network:

$$o = \frac{1}{1 + e^{-a}} \tag{8}$$

where

$$a = w_0 + \sum_{\ell=1}^{n} w_\ell x_\ell(r) \tag{9}$$

Weights $w_\ell$ are also trainable.

Supervised learning is based on the maximum likelihood principle as in many other neural network models. In the case of classification, the likelihood function has the form of a cross-entropy. Let $\mathcal{D} = \{(I^{(k)}, \overline{o}^{(k)}), k = 1, \cdots, K\}$ be a training set of labeled trees (where $\overline{o}^{(k)}$ denotes the class of $I^{(k)}$). Then the cost function is the negative log-likelihood:

$$J(\mathcal{D}) = -\left( \sum_{k=1}^{K} \overline{o}^{(k)} \log o^{(k)} + (1 - \overline{o}^{(k)}) \log(1 - o^{(k)}) \right) \tag{10}$$

$o^{(k)}$ being the output of the network after processing the $k$-th tree. Optimization is solved by gradient descent. In this case, gradients are computed by a special form of backpropagation on the feedforward network obtained by unrolling the state transition network according to the topology of the input tree $I$. The algorithm was first proposed in (Goller & Küchler, 1996) and is referred to as *backpropagation through structure* (BPTS). The main idea is depicted in Figure 5: On the left an example input tree is shown. On the right, the state transition network $f()$ is unrolled to match the topology of the input tree. The output network $g()$ is attached to the replica of $f()$ associated with the root. After recursion (4) is completed, the state vector $\boldsymbol{x}(r)$ at the root $r$ contains an *adaptive* encoding of the

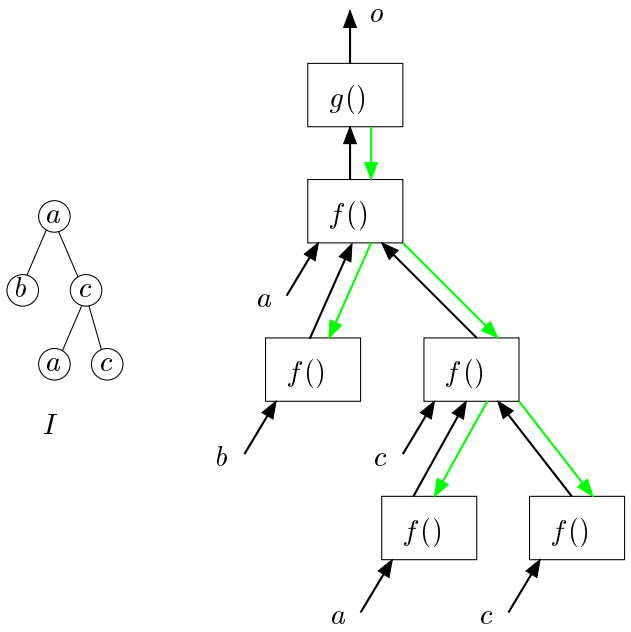



Figure 5: Network unrolling and backpropagation through structure

whole tree. This is similar to the encoding performed by recursive autoassociative memories first proposed by Pollack (Pollack, 1990) but here the encoding results from a supervised learning mechanism. Forward propagation (dark arrows) is bottom up and follows Eq. (4). Backward propagation (light arrows) is employed to compute the gradients and 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.

### 4.2 Recursive networks for ranking alternatives

As explained in Section 3, an alternative formulation of the learning task establishes that the input portion of each example is a forest of candidate trees. Such an object is not connected and thus not readily suitable as the input to a recursive neural network. In this formulation, moreover, we do not have a standard classification task. The network has to pick up one out of $m_i$ candidates, and the number of candidates depends on the particular instance (this is somewhat similar to having a variable number of classes). In this section we describe the required modifications to the basic model presented above, to make it suitable for solving the natural language learning task at hand.

The basic idea is rather simple and consists of processing each tree in the forest separately,

and then integrating results with a normalization step to satisfy Eqs. (2,3). As a first step, we modify the output function $g()$ (see Eqs. (8,9)) of the recursive model as follows:

$$o \quad = \quad w_0 + \sum_{\ell=1}^{n} w_\ell x_\ell(r)$$

i.e. the output network reduces to a (linear) weighted sum of the state components at the root of each tree. In this way we obtain a vector $\boldsymbol{o}_i = [o_{i,1}, \cdots, o_{i,m_i}]$ whose components are real numbers in $(-\infty, +\infty)$ associated with each candidate in $\boldsymbol{F}_i$. Correctness probabilities are finally obtained by taking the softmax function of $\boldsymbol{o}_i$:

$$y_{i,j} = \frac{e^{o_{i,j}}}{\sum_{\ell=1}^{m_i} e^{o_{i,\ell}}} \tag{11}$$

In this way we satisfy the multinomial model expressed by Eqs. (2,3) and $y_{i,j}$ can be interpreted as the probability that the $j$-th candidate tree in $\boldsymbol{F}_i$ is the correct tree. Each member of the forest is separately processed by this modified recursive network. The same network is used to process each tree in the forest so the resulting unrolled network is a forest of neural networks sharing the same weights for the transition function $f()$ and the output function $g()$. An immediate consequence of this choice is that the probabilities $y_{i,j}$ are not affected by a permutation of trees in the candidate set, a property which is clearly desirable at this level.

The cost function (negative log-likelihood according to the multinomial model) is written as follows:

$$J(\mathcal{D}) = -\sum_{p=1}^{P} \sum_{i=0}^{|s^{(p)}|-1} \log y_{i,j*} \tag{12}$$

where $\mathcal{D}$ denotes the training set, the first sum ranges over sentences in the training set, the second sum ranges over words within each sentence, and $j^\star$ is the index of the correct incremental tree in the candidate list (which is known in the training set).

At this point, the calculation of delta errors needed to compute gradients is straightforward. It begins by following the classic scheme of multinomial classification described in (Rumelhart et al., 1995), that we report here for completeness. Denoting

$$\delta_{i,j} \doteq \frac{\partial J(\mathcal{D})}{\partial o_{i,j}}$$

we have

$$\delta_{i,j} = \begin{cases} 1 - y_{i,j} & \text{if } j = j* \\ -y_{i,j} & \text{if } j \neq j* \end{cases} .$$

For each network $j$ in the forest, $\delta_{i,j}$ is injected as the delta output error, and subsequently passed to the BPTS algorithms. Of course, since networks in the forest share the same weights, gradients are computed by summing all the contributions relative to the same shared weight.

# 5 Implementation and results

## 5.1 Data preparation and experimental setup

The experiments were run on a set $\boldsymbol{B}$ of 2000 parsed sentences, randomly extracted from the Wall Street Journal Section of the Penn II Treebank. The universe of paths $U(\boldsymbol{B})$ was extracted from these sentences using the incremental parsing simulation algorithm described in Section 2. This resulted in a set of 1,675 connection paths, that we kept indexed by anchor and foot. Then, we selected four subsets of $\boldsymbol{B}$, called $\boldsymbol{B}_1$, $\boldsymbol{B}_2$, $\boldsymbol{B}_3$, and $\boldsymbol{B}_4$, containing 100, 500, 100, and 500 sentences, respectively. $\boldsymbol{B}_2$, $\boldsymbol{B}_3$, and $\boldsymbol{B}_4$ were disjoint, and were used to construct the training set, the validation set[5], and the test set, respectively. $\boldsymbol{B}_1$ is a subset of $\boldsymbol{B}_2$, and was also used as a training set. Each sentence in these sets was processed to extract a forest of candidate incremental trees as follows. A connection path is valid for $T_i$ if its anchor and its foot match the corresponding entities in $T_i$. Therefore, for every incremental tree we have to identify an anchor (that can be any node on the right frontier of $T_{i-1}$) and a foot (given by the rightmost leaf of $T_i$). At this point, the universe $U(\boldsymbol{B})$ is searched for legal paths matching every pair anchor-foot (see Figure 3). In turn, legal connection paths are then joined to $T_{i-1}$ to form the forest $\boldsymbol{F}_i$. On the available data, the average cardinality of a candidate set $\boldsymbol{F}_i$ is 56, while the maximum cardinality is 388. In the training set, $\boldsymbol{B}_1$, the total number of words is 2,685 and the total number of candidates is 183,841. In the test set, the total number of words is 11,617 and the total number of candidates is 698,735.

Nodes are labeled with non terminal symbols (syntactic categories) of the context free grammar used in the Penn II Treebank. We have decided to use a subset of non terminal symbols in order to filter out information that is not relevant to our task, or that would make some rules to be learned based on specific and rare examples. As a result, we reduced the nonterminal symbols to a subset of 71 elements, collapsing some of the non terminal

[5]The validation set was employed to estimate generalization performance, in order to stop gradient descent and prevent overfitting.

| ADJP | NP | PP |
|---|---|---|
| ADJP-ADV | NP-ADV | PP-BNF |
| ADJP-CLR | NP-BNF | PP-CLR |
| ADJP-PRD | NP-CLR | PP-DIR |
| ADVP-CLR | NP-DIR | PP-DIR-CLR |
| ADVP-DIR | NP-EXT | PP-DTV |
| ADVP-DIR-CLR | NP-HLN | PP-EXT |
| ADVP-EXT | NP-LGS | PP-LGS |
| ADVP-LOC | NP-LOC | PP-LOC |
| ADVP-LOC-CLR | NP-LOC-PRD | PP-LOC-CLR |
| ADVP-LOC-PRD | NP-MNR | PP-LOC-PRD |
| ADVP-MNR | NP-PRD | PP-MNR |
| ADVP-PRP | NP-SBJ | PP-PRD |
| ADVP-PUT | NP-TMP | PP-PRP |
| ADVP-TMP | NP-TTL | PP-PUT |

Table 1: Reduction of non terminal symbols. Entries within each column are merged into the symbol reported in the header of the column.

variations into their base form. Table 1 shows details of the merge we have performed.

The first recursive network used for the experiments implements the model described in Section 4.1, and has the following specifications: $m = 14$ (maximum outdegree), $N = 71$ labels (one-hot encoded), and $n = 40$ state components, yielding a total of about 25,000 adjustable parameters. The dimension $n$ must be large enough to give sufficient expressive power to the network, but not too large to avoid overfitting. In our experiments, $n = 40$ was sufficient to load the entire training set and to give good generalization accuracy. In this experiment we used $\boldsymbol{B}_1$ (100 sentences) a the training set. When using 0-1 classification, the ratio of positive over negative examples turns out to be very small (2,685/181,156 = 0.015). This follows from the fact that for each positive example there is a large number of competing candidate incremental trees. The network is trained to minimize the number of misclassifications, but this goal would be easily achieved by predicting always 0 (regardless of the input) since the base accuracy[6] for this learning problem is 98.5%. This theoretical

[6]The base accuracy is the fraction of correctly classified instances when always predicting the most frequent class.

expectation has been practically verified, and our experiments using all the available examples in each epoch did not produce any useful results (the network output was always very close to zero). We found that a viable way of dealing with this problem is to present the negative examples in the training phase with a lesser frequency than the positive ones, in order to re-establish an average ratio of 1:1 positive-negative examples per training epoch. The set of negative trees was rotated at each epoch, so any given negative tree is used once in a round of 67 epochs.

In a second experiment we implemented the multinomial model described in Section 4.2. In this case, each tree in the forest is processed separately by a recursive network with one linear output unit. The resulting numbers are then normalized with the softmax function. The recursive network used in the experiment had the same size of the one described above (i.e., about 25,000 weights). The training set was also $\boldsymbol{B}_1$. In this case, since the input to the network is an entire forest of candidates, the unbalance problem of positive/negative examples does not occur. All the trees (both the correct and the incorrect ones) are grouped into 2,685 forests and are employed to train the network in each epoch.

A third experiment was identical to the previous onw, but using $\boldsymbol{B}_2$ (500 sentences) as a training set. The experiment was performed to investigate the effect of the training set size on the prediction accuracy.

In all the experiments, we estimated generalization (the average position $R$ of the correct candidate) by running the partially trained networks in recall mode on the 100 validation sentences $\boldsymbol{B}_3$. Gradient descent was stopped when the performance on the validation set was maximum. This early stopping is expected to be useful in order to prevent overfitting. In the case of 0-1 classification, training was stopped after 1000 epochs. In the case of multinomial ranking, 90 epochs were sufficient (note, however, that less trees per epoch are processed in the first case, due to negative examples subsampling). For all the experiments, the reported accuracy (see Section 5.3) was measured on the test set $\boldsymbol{B}_4$ (500 sentences).

### 5.2 Psycholinguistic preferences

A fair comparison with other approaches to score parse decision can involve the psycholinguistic preferences mentioned in the introduction, Late Closure (LC) and Minimal Attachment (MA), that are at work in many parsing models because of their immediate implementation and the reasonable results they yield. Briefly, LC is the preference for attaching a new constituent at the lowest legal site on the right edge; MA is the preference

for the attachment that involves the creation of the minimum number of new nodes.

LC preference explains the surprise effect in the interpretation of a sentence. For example, consider "The president declared that he will resign yesterday." According to LC, "yesterday" is initially attached to the lower verb "will resign," but this attachment is then reject on the basis of the semantic knowledge ("yesterday" is incompatible with the future tense of "will resign"); on the contrary, the higher attachment to "declared" is semantically plausible. MA expresses a preference for flatter structures. In "John wrote a letter to Mary," "to Mary" is preferably attached to "wrote" rather than to "a letter," because of a minor number of nodes involved. Notice that in this case MA overrides LC, which would predict an attachment to "the letter."

Though it is common in the psycholinguistic literature to describe initial parsing decisions as the result of the application of a number of preferences, (almost) everybody does still believe that preferences for recency (as in LC) and structural simplicity (as in MA) have an influence on parsing decisions in some form or other. The reason for considering LC and MA in the context of this paper is that such preferences are of exclusively syntactic nature, and (though overridden by lexical and semantic heuristics) they are invoked when other sophisticate methods do not produce any valuable result. For example, Weischedel et al. (Weischedel et al., 1993) report a 75% of correctness of LC preference on 166 PP-attachment cases in the Penn treebank; these cases remained unsolved after a run of a partial parser based on a probabilistic model, and were accounted for by the LC preference[7]. Lombardo and Sturt (Lombardo & Sturt, 1999) report on a sophisticate form of MA in the context of a strongly incremental parser. The sophistication is due to the distinction of the syntactic nodes between headed and headless: headed nodes are those nodes for which the head word have already been scanned; otherwise, they are headless. Headedness reformulates MA from the number of nodes in a constituent to the number of headless nodes in a constituent: this reformulation is more adequate with the recent lexicalist trend in linguistics. According to this trend, syntactic phenomena cannot be adequately explained without including lexical knowledge in a theory of natural language syntax. For example, a rule of a context free grammar for natural language may exist only if there exists a word (in the lexicon) that licenses it. The results on a sample of about 50,000 words from the Wall Street Journal section of the Penn treebank were that the 82% of connection paths had 0 headless nodes, 15% had 1 headless nodes, 2% had 2 headless nodes, and 1% had 3 headless nodes.

[7]Note that this 75% correctness cannot be compared with the results described in this paper (Section 5.3), since they are based on two different parsing models.

Since LC and MA are often in conflict with each other (see example "John wrote a letter to Mary" above), we have collected the data for the two preferences by alternating the one which overrides the other: in one trial, LC overrides MA (LC over MA), that is, incremental trees are ranked higher if the connection path anchors low and then with a minimum number of nodes; in the other trial, MA overrides LC (MA over LC), that is, incremental trees are ranked higher if they have a minimum number of nodes and then the connection path anchors low. In the next section we report the results.

### 5.3 Results

After training, the network assigns a probability to each candidate incremental tree. A parser can employ the network output to rank candidate trees (by increasing probability of correctness) for continuation. Hence, the performance measure of our approach to parse decisions is based on the rank assigned to the correct tree in each forest $\boldsymbol{F}_i$.

A synthesis of results on the 500 test sentences is reported in Table 2. In the table, $R$ is the average position of the correct tree, after having sorted candidates according to network predictions. $R = 1$ would yield a perfect predictor that always ranks the correct tree in first position. The next column, $F$, is the percentage of times the correct tree is ranked in first position. Global results are reported in the first row of the table. We can see that the multinomial ranking ("ANN multin.") outperforms 0-1 classification by about one position, when both models are trained on $\boldsymbol{B}_1$ (100 sentences). Of course, when the number of candidates is higher, we can expect that it is more difficult to rank the correct one on the first positions. To give a more detailed overview of prediction quality, we should report $R$ separately for each value of $\boldsymbol{F}_i$. For conciseness, results are averaged over bins associated with ranges of candidate list lengths. The total number of forests in each bin is also reported in the last column. Multinomial ranking further improves when training on 500 sentences (set $\boldsymbol{B}_2$). The results on the mean value $R$ clearly demonstrate that the ranking produced by the network significantly reduces the number of candidate trees at each incremental parsing step. For example, when the number of candidates is between 100 and 111, the correct tree is ranked in position 1.95 (on average).

Columns 8–11 in Table 2 report $R$ and $F$ for the two strategies (LC over MA, and MA over LC) described in Section 5.2. The results obtained with the neural network model compare very favorably with respect to psycholinguistic preferences. The only exception is when there are only 2 or 3 alternative trees (in this case MA and LC preferences slightly

| | ANN multin. trained on 500 | | ANN multin. trained on 100 | | ANN 0-1 trained on 100 | | LC over MA | | MA over LC | | # of test |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $\lvert \boldsymbol{F}_i \rvert$ | $R$ | $F$ | $R$ | $F$ | $R$ | $F$ | $R$ | $F$ | $R$ | $F$ | forests |
| 2–388 | 2.02 | 74.0% | 3.26 | 63.5% | 4.29 | 31.6% | 4.74 | 48.8% | 6.90 | 42.2% | 11,617 |
| 2–3 | 1.17 | 85% | 1.25 | 77% | 1.41 | 63% | 1.35 | 75% | 1.36 | 75% | 381 |
| 4–5 | 1.20 | 88% | 1.54 | 60% | 2.05 | 33% | 2.62 | 28% | 2.48 | 25% | 427 |
| 6 | 1.41 | 82% | 1.92 | 59% | 2.10 | 38% | 2.90 | 29% | 2.67 | 27% | 168 |
| 7 | 1.52 | 75% | 1.85 | 50% | 2.29 | 33% | 3.17 | 32% | 3.01 | 29% | 275 |
| 8 | 1.27 | 87% | 1.49 | 76% | 1.81 | 65% | 2.81 | 51% | 2.77 | 48% | 274 |
| 9 | 1.52 | 74% | 2.45 | 34% | 2.42 | 44% | 4.00 | 37% | 3.54 | 36% | 221 |
| 10 | 1.31 | 82% | 1.79 | 60% | 2.54 | 43% | 3.04 | 42% | 3.02 | 39% | 273 |
| 11–13 | 1.50 | 78% | 2.31 | 56% | 2.44 | 49% | 5.27 | 30% | 4.64 | 19% | 308 |
| 14–16 | 1.51 | 79% | 1.88 | 65% | 2.41 | 47% | 4.03 | 41% | 3.97 | 36% | 380 |
| 17 | 1.84 | 73% | 2.29 | 49% | 2.97 | 30% | 4.09 | 21% | 4.25 | 17% | 219 |
| 18 | 1.21 | 92% | 1.64 | 79% | 1.74 | 77% | 1.99 | 79% | 2.20 | 76% | 298 |
| 19–20 | 1.90 | 73% | 2.88 | 50% | 3.09 | 28% | 4.22 | 34% | 3.79 | 28% | 258 |
| 21–23 | 1.47 | 76% | 2.56 | 48% | 3.05 | 28% | 4.70 | 42% | 4.47 | 36% | 386 |
| 24–27 | 1.80 | 69% | 3.25 | 47% | 3.48 | 31% | 5.10 | 44% | 4.39 | 39% | 402 |
| 28–30 | 1.88 | 74% | 2.29 | 54% | 2.80 | 26% | 4.36 | 39% | 4.36 | 28% | 381 |
| 31–34 | 2.13 | 63% | 3.12 | 49% | 3.75 | 29% | 4.69 | 43% | 4.83 | 38% | 287 |
| 35–38 | 1.74 | 70% | 2.99 | 52% | 3.70 | 32% | 4.72 | 41% | 5.56 | 36% | 425 |
| 39–41 | 2.38 | 58% | 3.14 | 51% | 4.05 | 24% | 6.68 | 33% | 7.47 | 27% | 294 |
| 42–45 | 1.91 | 75% | 2.41 | 59% | 3.16 | 32% | 4.22 | 48% | 5.37 | 43% | 355 |
| 46–49 | 1.89 | 76% | 2.67 | 54% | 3.45 | 32% | 4.56 | 54% | 5.59 | 46% | 319 |
| 50–53 | 2.31 | 63% | 3.34 | 53% | 4.16 | 27% | 4.18 | 45% | 5.79 | 36% | 372 |
| 54–58 | 1.97 | 75% | 2.84 | 52% | 3.44 | 30% | 4.29 | 61% | 5.30 | 54% | 420 |
| 59–64 | 1.87 | 72% | 3.72 | 51% | 4.40 | 31% | 3.47 | 62% | 5.29 | 53% | 357 |
| 65–69 | 2.50 | 69% | 3.57 | 50% | 5.21 | 19% | 5.89 | 45% | 8.30 | 38% | 303 |
| 70–74 | 2.43 | 69% | 5.39 | 49% | 5.82 | 24% | 5.61 | 56% | 6.45 | 50% | 385 |
| 75–81 | 2.02 | 72% | 3.31 | 57% | 4.52 | 27% | 4.93 | 54% | 7.95 | 44% | 414 |
| 82–89 | 1.78 | 71% | 4.01 | 46% | 5.26 | 21% | 5.95 | 47% | 10.42 | 38% | 360 |
| 90–99 | 2.42 | 70% | 4.29 | 50% | 5.51 | 23% | 4.24 | 59% | 8.03 | 50% | 386 |
| 100–111 | 1.95 | 74% | 4.17 | 42% | 5.19 | 23% | 5.86 | 54% | 9.00 | 42% | 444 |
| 112–126 | 3.06 | 71% | 4.80 | 51% | 6.90 | 20% | 6.20 | 49% | 10.45 | 39% | 427 |
| 127–145 | 2.53 | 69% | 4.44 | 49% | 5.85 | 20% | 6.19 | 49% | 12.72 | 37% | 374 |
| 146–173 | 3.42 | 67% | 4.76 | 42% | 7.33 | 17% | 7.69 | 43% | 13.94 | 31% | 381 |
| 174–222 | 3.23 | 61% | 7.14 | 33% | 11.21 | 14% | 8.83 | 36% | 18.65 | 26% | 395 |
| 223–388 | 5.64 | 56% | 7.02 | 32% | 13.60 | 11% | 13.46 | 26% | 32.75 | 19% | 268 |

Table 2: Prediction performance (see text for explanation).

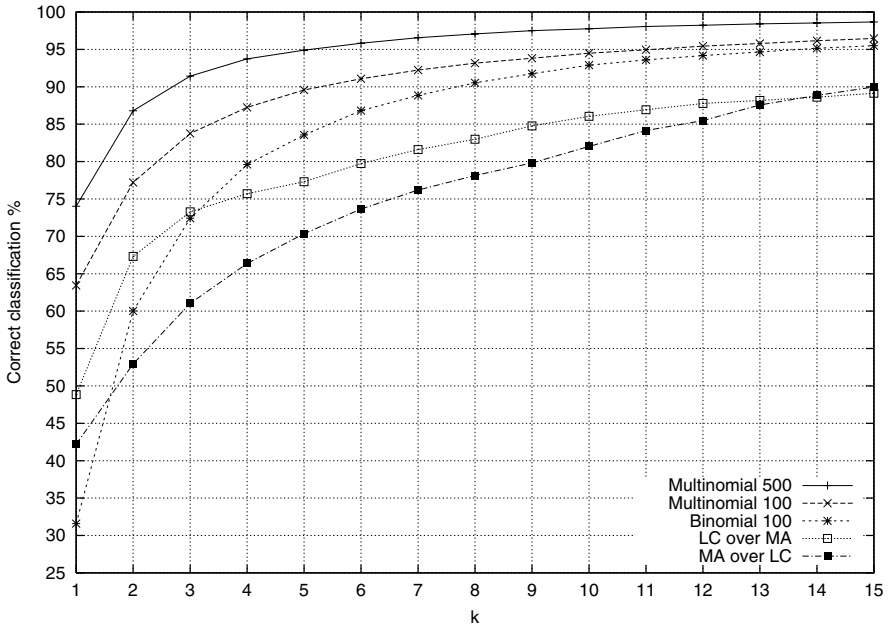



Figure 6: Experimental comparison of knowledge-based heuristics (Late Closure, Minimal Attachment) and neural network predictions (Multinomial ranking and 0-1 classification). The graph reports the percentage of times that, after sorting candidates by one of the heuristics, the correct tree is found within the first $k$ trees, for $k = 1, \ldots, 20$.

outperform the neural network). Moreover, multinomial ranking performs significantly better than the 0-1 approach. This can be expected theoretically since the architecture proposed in Section 4.2 is a more accurate model of the task at hand. This can also be explained intuitively: if the correct candidate is not ranked in first position, the contribution to cost function (12) is large. On the other hand, the contribution to cost function (10) could be small if the score of the correct candidate is close to 1 but other competing candidates achieve higher scores. This explains why $F$ is consistently higher for the multinomial ranking method.

In Figure 6 we report the cumulative histogram of the five ranking models. Two histograms are reported for the multinomial method, one using training set $\boldsymbol{B}_1$ (100 sentences), and another using training set $\boldsymbol{B}_2$ (500 sentences). Test is done on $\boldsymbol{B}_4$ in all the five cases. The graph represents the percentage of times that the correct tree is ranked within the first $k$ positions. Again, we can see that neural network based predictions outperform the two heuristics.

### 5.4 Discussion

Although too weak for a complete parsing model, the predictions summarized in Table 2 and Figure 6 are very encouraging, and certainly motivate a research on the employment of incremental parsing supported by RNN decisions as an effective model for wide coverage parsing.

The results reported here need some comments in the NLP setting. A general comment concerns the confirmation in our data of the complexity of natural language parsing. In the strongly incremental version adopted in this work, where the problem is coded as the selection of the best incremental tree, the data on the Penn treebank sample show that the search space has an average branching factor of 56, with a maximum factor of 388. These data confirm the necessity of having some form of sophisticate elaboration of parse decisions.

The rank position of the correct incremental tree is consistently low in the candidate list. The result is particularly positive, especially if taken in its naive conception. Consider that corpus-based models in the literature usually take into account several cues for reaching a parse decision; on the contrary, in the experiments described here only syntactic knowledge is taken into account. Lexical knowledge, which is at the core of many parsing models (see, e.g., (Srinivas & Joshi, 1999)), is totally neglected here. Connection paths include at most POS-tags. On the contrary, most models are trained on relations established between the head words of syntactic phrases; the use of syntactic categories is limited to those cases when data about lexical relations are sparse. Also semantics is taken into account when available for limited domains, and including subcategorization frames and a number of thematic roles. The II version of the Penn treebank includes the latter knowledge because of the relevance for disambiguation.

Actually, some of the structural ambiguities on which the network performs badly are the PP-attachment cases. It is exactly on these cases that corpus-based models employ the lexical and semantic knowledge. Other common errors are then due to the lack of context for decisions concerning initial words and to the large extension of the context (right frontier) for decisions concerning words at the end of a sentence. The first typology of errors could be dealt with by delaying initial parse decisions for the first word in the sentence; the second typology could hardly be addressed without introducing lexical knowledge. For other types of errors, we did not found an obvious relationship with grammatical structure.

If structural preferences are not enough for a complete parsing model, their contribution to human parsing decisions needs to be well assessed. A first result in this direction is an experiment on the validation of a number of psycholinguistic preferences in a realistic environment. In the psycholinguistic community, it has often been claimed that ambiguity resolution involves structural preferences (as is the case for Late Closure and Minimal Attachment seen above), and this has been shown on toy grammars. However, a precise characterization of structural preferences is hard to achieve as soon as one begins to work with realistically large grammars. A wide-coverage model of structural preferences in the human parser must then rely on preference estimation from a corpus: the model must be able to work with tree structures that are large enough to represent any structural ambiguity, but should also be able to deal with the sparse-data problem, as the larger the configuration, the less frequently it will be found in any corpus. So, the model described in this paper has been applied to the prediction of first-pass attachment decisions on psycholinguistic examples (Sturt, Lombardo, Costa, & Frasconi, 2001; Sturt, Costa, Lombardo, & Frasconi, 2001). First-pass attachment decisions concern the disambiguation that takes place as soon as a one hears/reads a passage; this decision then undergoes a revision when the structure that has been chosen does not fit with subsequent material. So, the goal is not to predict the correct parsing decision with respect to the whole sentence, but to reproduce a human preference which is susceptible of revision. Using the training corpus described in Section 5.1, the model reproduces correctly a number of well known preferences, such as recency attachment for adverbs (a sort of Late Closure limited to adverbs), relative clause attachment in English, the preference for NP over S in complement ambiguities (in sentences like "The athlete realized his goals" vs. "The athlete realized his goals were out of reach"). If we want to pursue further the model of the human parser, we must provide a correct way to recover from first-pass attachment errors.

# 6 Conclusions

The paper has presented a novel methodology for parsing unrestricted texts, based upon the incrementality hypothesis, a widely held assumption about the human parser. A recursive neural network, trained on tree portions extracted from the Penn treebank via a simulation of the incremental strategy, carries out the parse decisions in case of ambiguity. The decision is in the form of the selection of the correct incremental tree in the list of possible candidates for continuation in left-to-right processing. The learning task is formulated as

the identification of the correct incremental tree in the list of candidates; the prediction is a ranking of the candidates according to the conditional probability of being the correct one. The results, although the method currently takes into account only the syntactic structure labeled with non terminal categories, are very positive. The comparison with well-known parsing preferences in the psycholinguistic literature clearly indicate a superiority of the parse predictions provided by the network. On the architectural side, a novel contribution of this paper is the proposal of a neural network approach for learning to rank candidates. The same architecture can, in principle, be applied to other tasks in which each object is a collection of instances among which a "winner" is to be probabilistically selected.

The future application of our results to parsing are immediate, and consists in the development of an efficient incremental parser which is informed by the network architecture in taking decisions about attachment ambiguity. An immediate improvement to the methodology presented here is the insertion of other features in the definition of the learning task. Corpus-based approaches usually rely on lexical and thematic knowledge for the formulation of parse decisions. Purely syntactic knowledge is empirically demonstrated to be insufficient in a realistic parsing process. Moreover, a clear direction to pursue further the model of the human parser is to provide a correct way to recover from first-pass attachment errors, possibly including non-local information as done for example in (Roark & Johnson, 1999).

### Acknowledgment

We would like to thank Patrick Sturt for providing the processed corpus of sentences used in the experiments, and for comments on an early draft of this paper.

## References

Auer, P. (1997). On learning from multi-instance examples: Empirical evaluation of a theoretical approach. In Fisher, D. H. (Ed.), *Proc. 14th Int. Conf. Machine Learning*, pp. 21–29. Morgan Kaufmann.

Bader, M., & Lasser, I. (1994). German verb-final clauses and sentence processing. In Clifton, C., Frazier, L., & Reyner, K. (Eds.), *Perspectives on Sentence Processing*, pp. –. Lawrence Erlbaum Associates.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Bianucci, A., Micheli, A., Sperduti, A., & Starita, A. (2000). Application of cascade-correlation networks for structures to chemistry. *Applied Intelligence*, *12*, 115–145.

Collins, M. (1996). A new statistical parser based on bigram lexical dependencies. In *Proc. of 34th ACL*, pp. 184–191.

Dietterich, G., Lathrop, R. H., & Lozano-Perez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*(1–2), 31–71.

Eberhard, K. M., Spivey-Knowlton, M. J., Sedivy, J., & Tanenhaus, M. K. (1995). Eye movements as a window into real-time spoken language comprehension in natural contexts. *Journal of Psycholinguistic Research*, *24*, 409–436.

Fodor, J. D., & Ferreira, F. (Eds.). (1998). *Reanalysis in sentence processing.* Kluwer Academic Publishers.

Francesconi, E., Frasconi, P., Gori, M., Marinai, S., Sheng, J., Soda, G., & Sperduti, A. (1997). Logo recognition by recursive neural networks. In Kasturi, R., & Tombre, K. (Eds.), *Graphics Recognition – Algorithms and Systems*. Springer Verlag.

Frasconi, P., Gori, M., & Sperduti, A. (1998). A general framework for adaptive processing of data structures. *IEEE Trans. on Neural Networks*, *9*(5), 768–786.

Frazier, L. (1987). Syntactic processing: Evidence from dutch. *Natural Language and Linguistic Theory*, *5*, 519–559.

Frazier, L., & Fodor, J. D. (1978). The sausage machine: A new two-stage parsing model. *Cognition*, *6*, 291–325.

Goller, C. (1997). *A Connectionist Approach for Learning Search-Control Heuristics for Automated Deduction Systems.* Ph.D. thesis, Tech. Univ. Munich, Computer Science.

Goller, C., & Küchler, A. (1996). Learning task-dependent distributed structure-representations by backpropagation through structure. In *IEEE International Conference on Neural Networks*, pp. 347–352.

Hermjakob, U., & Mooney, R. J. (1997). Learning parse and translation decisions from examples with rich context. In *Proceedings of ACL97*, pp. 482–489.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, *46*, 47–75.

Hobbs, J., & Bear, J. (1990). Two principles of parse preference. In *Proceedings of COLING90*, pp. 162–167.

Kamide, Y., & Mitchell, D. C. (1999). Incremental pre-head attachment in japanese parsing. *Language and Cognitive Processes*, *14*(5–6), 631–662.

Kimball, J. P. (1973). Seven principles of surface structure parsing in natural language. *Cognition*, *2*, 15–47.

Kolen, J., & Kremer, S. (Eds.). (2000). *A Field Guide to Dynamical Recurrent Networks*. IEEE Press.

Lombardo, V., Lesmo, L., Ferraris, L., & Seidenari, C. (1998). Incremental processing and lexicalized grammars. In *Proceedings of the XXI Annual Meeting of the Cognitive Science Society*, pp. 621–626.

Lombardo, V., & Sturt, P. (1999). Incrementality and lexicalism: A treebank study. In Stevenson, S., & Merlo, P. (Eds.), *Lexical Representations in Sentence Processing*. John Benjamins.

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

Marslen-Wilson, W. (1973). Linguistic structure and speech shadowing at very short latencies. *Nature*, *244*, 522–533.

Milward., D. (1995). Incremental interpretation of categorial grammar. In *Proceedings of EACL95*.

Nagao, M. (1994). Varieties of heuristics in sentence processing. In *Current Issues in Natural Language Processing: In Honour of Don Walker*. Giardini with Kluwer.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623–641.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, *46*(1-2), 77–106.

Roark, B., & Johnson, M. (1999). Efficient probabilistic top-down and left-corner parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pp. 421–428.

Rumelhart, D. E., Durbin, R., Golden, R., & Chauvin, Y. (1995). Backpropagation: The basic theory. In *Backpropagation: Theory, Architectures and Applications*, pp. 1–34. Lawrence Erlbaum Associates, Hillsdale, NJ.

Sperduti, A., & Starita, A. (1997). Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, *8*(3).

Srinivas, B., & Joshi, A. (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, *25*(2), 237–265.

Stabler, E. P. (1994). The finite connectivity of linguistic structure. In Clifton, C., Frazier, L., & Reyner, K. (Eds.), *Perspectives on Sentence Processing*, pp. 303–336. Lawrence Erlbaum Associates.

Steedman, M. J. (1989). Grammar, interpretation and processing from the lexicon. In Marslen-Wilson, W. M. (Ed.), *Lexical Representation and Process*, pp. 463–504. MIT Press.

Sturt, P., Costa, F., Lombardo, V., & Frasconi, P. (2001). Learning first-pass attachment preferences with dynamic grammars and recursive neural networks. In preparation.

Sturt, P., & Crocker, M. (1996). Monotonic syntactic processing: a cross-linguistic study of attachment and reanalysis. *Language and Cognitive Processes*, *11*(5), 449–494.

Sturt, P., Lombardo, V., Costa, F., & Frasconi, P. (2001). *A wide-coverage model of first-pass structural preferences in human parsing.* 14th Annual CUNY Conference on Human Sentence Processing, Philadelpha, PA.

Thacher, J. (1973). Tree automata: An informal survey. In Aho, A. (Ed.), *Currents in the Theory of Computing*, pp. 143–172. Prentice-Hall Inc., Englewood Cliffs.

Weischedel, R., Meter, M., Schwartz, R., Ramshaw, L., & Palmucci, J. (1993). Coping with ambiguity and unknown words through probabilistic models. *Computational Linguistics*, *19*(2), 359–382.

Yamashita, K. (1994). *Processing of Japanese and Korean.* Ph.D. thesis, Ohio State University, Columbus, Ohio.