2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Integrating Bi-directional Contexts in a Generative Kernel for Trees

Davide Bacciu, Alessio Micheli and Alessandro Sperduti

Abstract— **Context is essential to evaluate an atomic piece of information composing an articulated structured sample. A particular context captures different structural information with respect to an alternative context. The paper introduces a generative kernel that easily and effectively combines the structural information captured by generative tree models characterized by different contextual capabilities. The proposed approach exploits the idea of hidden states multisets to realize a tree encoding that takes into account both the summarized information on the path leading to a node (i.e. a top-down context) as well as the information on how substructures are composed to create a subtree rooted on a node (bottom-up context). An thorough experimental analysis is provided, showing that the bi-directional approach incorporating top-down and bottom-up contexts yields to superior performances with respect to the unidirectional contexts alone, achieving state of the art results on challenging tree classification benchmarks.**

I. Introduction

STRUCTURED data models compound information such that a relevant part of the informative content is captured by the structural relationships between the atomic entities composing the sample. Such structural relationships, in a sense, provide a context in which the piece of information needs to be evaluated; different classes of structures (sequences, trees or more general graphs) entail different complexities and richness of such context. Dealing with such information becomes a matter of being able to effectively and efficiently capture the correlation between atomic pieces of data and their context, but also a matter of choosing what is the most appropriate and effective context. For instance, an element (node) of a sequence, that is the simplest structure type, can be evaluated in the context of either its predecessor or successor node. Similarly, a node label in a tree-structured piece of information may be evaluated in the context of either its surrounding descendants or ancestors.

The choice of the context ultimately determines how the structured sample is parsed and processed by a machine learning model for structured data. In this paper, we address the question of whether the choice a particular context entails the capability of capturing different structural information with respect to an alternative context and, in particular, how such different representational capability can be combined and exploited to yield more discriminative learning models for structured data.

The intuition that knowledge from different structural contexts can be integrated to increase the accuracy of a learning system has been discussed in [1] for sequential data. In [1], it is observed that information from both the past and the future portions of a finite sequence can be very useful for analysis and predictions at a given position (or time) t. This is the case, for instance, of DNA sequences where the function of a sequence region may strongly depend on observations located both upstream and downstream of the region. To this end, the authors in [1] propose a bi-directional generative approach such that the sequence is parsed both from left-to-right (i.e. predecessor context) as well as from right-to-left (i.e. successor context).

In this work, we address the integration of bi-directional contexts in a more complex class of structures, that is tree-structured data. Moving from the sequential to the tree domain introduces differences in the expressive power of the computational models which depend on the direction in which the structure is being parsed, i.e. the context in which its constituents are processed. From automata theory it is well known, for instance, that processing of a sequence left-to-right or right-to-left is equivalent, whereas the tree language recognizable by a deterministic top-down automaton (i.e. processing the tree from root to the leaves) is a strict subset of that recognized by the bottom-up counterpart (i.e. processing the tree from the leaves to the root) [2]. Such differences have a considerable impact on probabilistic learning models for structured data, where the choice of the context determines the direction of the probabilistic generative process and may change the associated probabilistic assumptions. Consider a hidden Markov model (HMM) [3] in the sequential domain: the inversion of the sequence parsing direction changes the orientation of the causal relationships in the model but the two dynamic Bayesian Networks (DBNs) originated by the two different contexts are equivalent from the point of view of the Markov properties [4]. A similar generative model for trees, on the other hand, is more deeply influenced by a change in the context, such that bottom-up [5], [6] and top-down [7], [8] approaches are characterized by different causal relationships that produce two different DBNs from a Markov-equivalence point of view [6]. This creates differences in the local Markov properties of the two models which influence the way the nodes exchange information during inference and learning [4], and ultimately determine what structural information is captured by the hidden states associated to the tree nodes. These hidden states essentially

Davide Bacciu and Alessio Micheli are with the Dipartimento di Informatica, Università di Pisa Largo B. Pontecorvo 3, Pisa, Italy (email: {bacciu,micheli}@di.unipi.it).

Alessandro Sperduti is with the Dipartimento di Matematica, Università di Padova Via Trieste 63, Padova, Italy (email: sperduti@math.unipd.it).

This work is partially supported by the FP7 RUBICON project (contract n. 269914).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

summarize the information on the substructure that has been generated until the node they are associated to. In other words, the hidden state space of a bottom-up model provides a summarized view of the subtrees occurring in the data, where each hidden state identifies a cluster of similar structures. On the other hand, the top-down approach provides a summarized view where each hidden state clusters similar root-to-node paths. It has been shown that the introduction of a context including both predecessors and successors information for each node, when moving to structured domains, entails an increase in computational capabilities of the neural network models, see [9] for a theoretical analysis on contextual models for structures within recursive incremental neural networks approaches [10].

Motivated by this, we investigate an approach that allows combining the summarized information provided by the bottom-up and top-down directional contexts. Rather that formalizing a bi-directional generative process characterized by an high-dimensional parameterization and by a considerable computational complexity, we focus on a computationally effective, incremental approach that is capable of combining the information captured by the two independent generative models. We know this information to be summarized by the hidden Markov states: recently, [11] has proposed an adaptive kernel for generative tree models that exploits the information captured by multisets of their Markov hidden states. We exploit the same intuition to derive a bi-directional kernel which incorporates multiset information from the hidden states of an independent bottom-up and top-down model. An adaptive kernel defines a data-induced similarity measure upon which learners, e.g. support vector machines, are built to solve classification/regression problems.

The introduction of contextual information within kernels for trees have been explored, so far, through forms of positional matching between subtrees. The underlying intuition is that an effective tree similarity should weigh more those substructures presenting similar features in the same position of the tree. The Route kernel [12], for instance, is a non-adaptive kernel building on the concept of route, that is the shortest path linking two nodes in a tree, and that measures tree similarity in terms of number of common routes. In [13], it is presented a general approach, Position Aware Kernel (PAK), that allows extending non-adaptive convolutional kernels with positional features using the route approach. A similar method is applied in [14] to add route information to an adaptive kernel built on the top of a topographic mapping model for trees. The approach put forward in this paper is different, as it introduces the possibility of fusing the context from multiple adaptive models characterized by different capabilities in terms of structural information summarization.

Through an experimental assessment on real-word tree classification benchmarks, we study if and under which conditions the proposed bi-directional kernel yields to an increase in the classification performance with respect to the unidirectional top-down and bottom-up kernels, suggesting that the two generative models effectively capture different forms of structural information.

II. Exploiting Bi-Directional Context in Generative Kernels

The section introduces a bi-directional adaptive kernel founding on the use of two probabilistic learning models for trees. Section II-A introduces the top-down and bottom-up approaches to generative modeling of trees which are later used in Section II-B to devise the bi-directional context kernel founding on Jaccard multisets similarity.

A. Generative Models for Trees: Top-down and Bottom-up Approaches

Generative approaches for trees allow modeling probability distributions over spaces of trees. This is achieved by generalizing the HMM approach for the sequential domain, through learning of an hidden generative process for labeled trees, that is regulated by hidden state variables modeling the structural context of a node and determining the emission of its label. By borrowing the nomenclature from HMM, these models are typically referred to as *Hidden Tree Markov Models* (HTMMs).

To formalize the notation used throughout the paper, we consider a dataset $\mathcal{D} = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$ of N labeled rooted tree where $\mathbf{x}^n$ is a connected acyclic graph consisting of a set of nodes $\mathcal{U}_n = \{1, \dots, U_n\}$ such that a single vertex is denoted as the *root* and any two nodes are connected by exactly one simple path. The index n is used to denote the n-th tree in a dataset of N structures and will be omitted for notational simplicity when the context is clear. In the following, the term $u \in \mathcal{U}_n$ is used to denote a generic nodes of $\mathbf{x}^n$. The direct ancestor of a node u, i.e. the node which is directly connected to u, is called its *parent*. By definition, each node u has at maximum one parent, however it can have a variable number of direct descendants (*children*), such that the l-th child of node u is denoted as $ch_l(u)$. For the purpose of this paper, we assume trees to have a finite maximum *outdegree* L, i.e. the maximum number of children of a node. Finally, each vertex u in the tree is associated with a label x_u which is a d-dimensional vector.

As anticipated in Section I, the direction of the generative process determines the context in which a node is evaluated and influences the features of the probabilistic model, as it affects its underlying probabilistic assumptions. We consider two generative processes associated to top-down and bottom-up parsing directions. The *top-down* HTMM [7], [8] (TD, in the following), for instance, implements a generative process for all paths from the root to leaves of the trees. This is realized by a set of hidden state variables associated with a state transition dynamics that follows the direction of the generative process, i.e. from a node u towards its children $ch_l(u)$. Specifically, an observed tree $\mathbf{x}^n$ is modeled by a set of hidden state variables $\{Q_1, \dots, Q_u, \dots\}$ following the same indexing as the observed nodes $u \in \mathcal{U}_n$ and assuming values on the discrete set of hidden states $\{1, \dots, C\}$. The direction of the generative process is then modeled by the

state transition probability

$$P(Q_u = j | Q_{pa(u)} = i) \tag{1}$$

entailing that the hidden state of a node is conditionally independent of the rest of the tree once that its parent state is observed. To complete the specification of the model, it is assumed that the label x_u (continuous or discrete) of a node u is completely specified by its hidden state Q_u through the *emission distribution* $P(x_u|Q_u = j)$. Following such conditional independence relations, we can factorize the joint distribution of an observed tree $\mathbf{x}^n$ with the hidden states assignment $Q_1, \ldots, Q_{U_n}$ as

$$\begin{aligned} P(\mathbf{x}^n, Q_1, \ldots, Q_{U_n}) =& P(Q_1)P(x_1|Q_1) \\ & \prod_{u=2}^{U_n} P(x_u|Q_u)P(Q_u|Q_{pa(u)}), \end{aligned} \tag{2}$$

where $P(Q_1)$ is the *prior* state distribution for the root node ($u = 1$), given that it has no parent. The likelihood for the TD is obtained from (2) by marginalizing the unknown hidden state assignment and summing across all dataset (see [8] for details).

The *bottom-up* HTMM (BU) [5], [6] defines a generative process propagating from the leaves to the root of the tree, which allows nodes to collect dependency information from each child subtree. The BU implements a generative process that composes the child subtrees of each node in the tree in a recursive fashion. Similarly to (1), this is modeled by a joint *state transition probability*

$$P(Q_u = i | Q_{ch_1(u)} = j_1, \ldots, Q_{ch_L(u)} = j_L) \tag{3}$$

assuming that each node u is conditionally independent of the rest of the tree when the joint hidden state of its direct descendants $Q_{ch_l(u)} = j_l$ is observed. The problem with the formulation in (3) is that it becomes computationally impractical for trees other than binary, since the size of the joint conditional transition distribution is order of C^{L+1}, where L is the node outdegree. In [6], this has been addressed by introducing a scalable *switching parent* approximation that factorizes (3) as a mixture of L pairwise child-parent transitions. The resulting BU joint distribution is

$$\begin{aligned} P(\mathbf{x}^n, Q_1, \ldots, Q_{U_n}) = & \prod_{u' \in \mathcal{LF}_n} P(Q_{u'})P(x_{u'}|Q_{u'}) \\ & \prod_{u \in \mathcal{U}_n \setminus \mathcal{LF}_n} P(x_u|Q_u) \sum_{l=1}^{L} P(S_u = l)P(Q_u|Q_{ch_l(u)}) \end{aligned} \tag{4}$$

where we recall that $\mathcal{LF}_n$ denotes the set of leaves in tree $\mathbf{x}^n$ and the summation term corresponds to the factorization of (3) using the switching parent $S_u \in \{1, \ldots, L\}$. This is a latent variable, independent from $Q_{ch_l(u)}$, and whose distribution $P(S_u = l)$ measures the influence of the l-th children on a state transition to node u.

Learning the parameters of the TD and BU generative models is addressed as an Expectation-Maximization (EM) [15] problem applied to the log-likelihood of the models, completed with latent indicator variables that model the unknown hidden state (and switching parent) assignments. Details of the learning algorithms for the specific models can be found in the papers cited above. The resulting algorithm is a two-stage iterative procedure, known also as the Baum-Welch algorithm [16] in the context of Markov chains, which allows to efficiently compute a solution to the log-likelihood maximization problem. At the E-step, it estimates the posterior of the indicator variables introduced in the completed log-likelihood, while, at the M-Step, it exploits such posteriors to update the model parameters θ. Posterior estimation is the most critical part of the algorithm and can be efficiently computed by message passing upwards and downwards on the structure of the nodes' dependency graph [6], [17]: this procedure is an extension to trees of the Forward-Backward inference algorithm for HMMs on sequences [3].

The TD and BU approaches are characterized by different context propagation strategies, that are determined by the direction of the state transition function (generative process) which, in practice, induces different conditional independence relationships. Such diverse generative dynamics leads to different representational capabilities. Consider the representation of the DBN of the TD and BU for a given tree in Fig. 1. With a top-down approach (Fig. 1(b)), the hidden state Q_u assigned to node u captures information about path π_u leading to the node from the root. With a bottom-up context (Fig. 1(a)), on the other hand, the hidden state assignment Q_u summarizes information concerning structural properties of its descending subtrees τ_u. In other words, the hidden state space of BU provides a summarized view of the subtrees occurring in the data, where each hidden state identifies a cluster of similar structures, while TD provides a summarized view where each hidden state clusters similar root-to-node paths. In [6] it has been shown that the conditional dependence relationships introduced by the BU context allow, in general, to capture more discriminant details on the tree structures with respect to a TD context. Nevertheless, we are convinced that the TD approach models a different form of structural information with respect to BU and that the integration of the two approaches can yield to more accurate and effective learning models for trees. In the following section, we discuss a way of fusing such information by means of generative kernels.

B. Integrating Top-Down and Bottom-Up Context in a Jaccard Kernel for Trees

The TD and BU exploit the concept of hidden states to define a generative model for tree-structured data, using the latent variables Q_u to simplify the conditional probabilities underlying the model. These hidden states summarize structural information concerning tree components, providing an adequate context, e.g., for the emission of a node label. We expect the hidden states of TD and of BU to encode different forms of structural contexts. By exploiting the concept of hidden states *multisets* [11], we discuss a generative kernel that integrates the structural information of independently trained TD and BU models. Roughly, each tree is represented

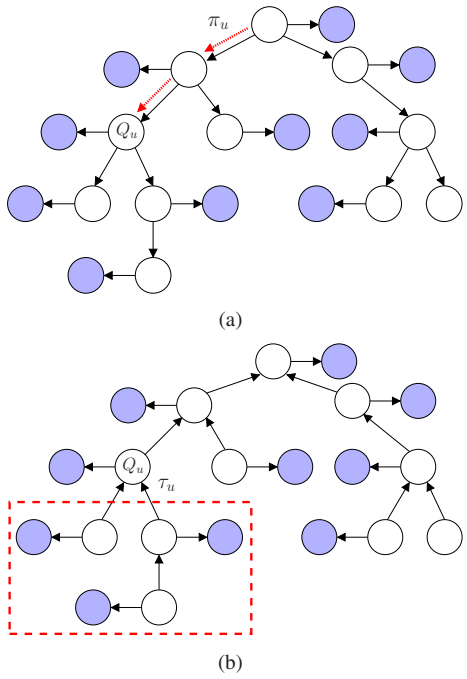

Fig. 1. Generative process associated to TD (a) and BU (b) models for a given tree: empty nodes denote latent variables (e.g. the hidden state Q_u) while shaded nodes are observable variables (e.g. the corresponding tree labels x_u); the switching parent nodes S_u are not shown in BU for the sake of simplicity. The figures highlight the context associated to an example node u in each model: π_u is the path leading to the node form the root; τ_u is the set of substructures rooted in u.

in terms of its associated hidden states both in the TD and BU models; then, structure similarity is computed on the basis of overlap in the hidden states' configurations. More specifically, given a trained TD and BU, we transform a tree $\mathbf{x}$ into a *bag-of-states*, that is a vector of hidden state counts, similarly to how textual documents are represented as vectors of word counts.

To compute the multiset encoding, we first need to determine the most likely hidden state assignment for a tree $\mathbf{x}$: this is referred to as the more general *decoding* problem in the literature on Hidden Markov Models [3]. Although there are different interpretations of optimality in the decoding problem, a widely accepted formulation is based on finding the hidden states that maximize the joint probability with the observation, i.e.

$$\max_{\mathbf{q}} P(\mathbf{X}=\mathbf{x}, \mathbf{Q}=\mathbf{q}|\theta), \tag{5}$$

where $\mathbf{x}$ is the observed tree, $\mathbf{q}$ is a (generic) associated hidden state assignment and θ is the set of parameters learned for BU (θ^{bu}) and TD (θ^{td}). The optimization problem in (5) is solved independently for the two generative models, obtaining the most likely hidden state assignment $Q^{td}_{n,u}$ and $Q^{bu}_{n,u}$ of each node u in tree $\mathbf{x}^n$ for TD and BU, respectively. This problem can be efficiently solved for both generative models through a dynamic programming approach, known as the *Viterbi Algorithm*, whose details can be found in [17] for TD and in [18] for BU.

Figure 2 describes the tree encoding process to obtain the bidirectional hidden states multiset. First, a sample tree (on the left) is associated to its Viterbi states in both the TD and BU models (right). By means of such Viterbi states, it is possible to define several bag-of-states encodings, depending on the amount of structural information that we want to introduce in the kernel feature-space representation: here, we consider two forms of bag-of-states, shown at the bottom of Fig. 2, corresponding to *unigram* and *bigram* multisets. The *unigram* is the simplest form of multiset that is based on counting the occurrence of the single hidden states (see left-side arrows in Fig. 2). Given a tree $\mathbf{x}^n$ and its associated TD hidden state assignments $Q^{td}_{n,u}$, it is transformed into a C-dimensional feature-vector $\Phi^{td}(\mathbf{x}^n)$ such that its i-th component is

$$\Phi^{td}_i(\mathbf{x}^n) = \sum_{u \in U_n} \delta(Q^{td}_{n,u}, i) \quad \text{and} \quad i = 1, \ldots, C \tag{6}$$

where we recall that $\mathcal{U}_n$ is the set of nodes in the n-th tree and $\delta(\cdot,\cdot)$ is the Kronecker function. The unigram encoding $\Phi^{bu}(\mathbf{x}^n)$ for the BU model is obtained as in (6). The unigram representation captures information on the prevalent *topics* in the tree, but does not convey any structural information, besides that captured by the generative model and conveyed by the hidden state assignment. To introduce more structural knowledge, we might be interested in modeling the co-occurrence of hidden-states in a parent-children relationship (see right-side arrows in Fig. 2). This is similar to when, in document analysis, we model the co-occurrence of two adjacent words in a text by means of a word bigram. In analogy to this, we define an hidden state *bigram*, where an input tree $\mathbf{x}^n$ is transformed in a (C^2)-dimensional feature-vector $\Phi^{td}(\mathbf{x}^n)$, such that its i_j-th element is

$$\Phi^{td}_{i_j}(\mathbf{x}^n) = \sum_{u \in U_n} \sum_{l \in ch(u)} \delta(Q^{td}_{n,u}, i)\delta(Q^{td}_{n,l}, j) \quad \text{and} \quad j = 1, \ldots, C,\ i_j = 1, \ldots, C \tag{7}$$

where $ch(u)$ is the set of children of node u (and $\Phi^{bu}(\mathbf{x}^n)$ is obtained similarly to (7)). The bigram encoding allows to represent the co-occurrence of hidden states patterns between a parent node and each of its children taken independently, thus providing the kernel with some form of (partial) structural information.

The unigram and bigram encodings can be computed by a single visit of the tree for each generative model. For efficiency, this calculation can be embedded in the steps of the Viterbi recursion of the underlying generative model, with only a minor modification in the (constants of the) Viterbi computational complexity. Algorithm 1 provides a procedural view of the steps needed to compute the encoding in BU (more details in [6]). For the purposes of Algorithm 1, we use the following notation: state transition $A^l_{ij} = P(Q_u = i|Q_{ch_l(u)} = j)$, prior probability $\rho_i = P(Q_u = i)$, switching parent distribution $\varphi_l = P(Q_u = l)$ and emission distribution $b_i(x_u) = P(x_u|Q_u = i)$. Note that the term ind in Algorithm 1 denotes an hash table whose element $ind(i, u, u') = j$ stores the Viterbi hidden state j associated

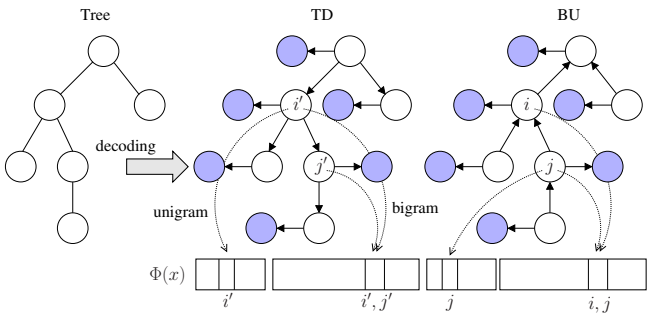

Fig. 2. Tree encoding process yielding to the bidirectional hidden states multiset: a test tree is associated to its most likely TD and BU hidden states through Viterbi decoding. Then unigram and bigrams are calculated for each generative model and concatenated in the final bidirectional representation. Note that, in the unigram representation (left-side arrow), the i-th vector component stores the number of occurrences of the i-th state in the tree. In the bigram (right-side arrow), there is a vector component for each pair of hidden states (e.g. i, j): the corresponding entry stores the co-occurrence counts of the first hidden state (e.g. i) being associated to a node whose child is assigned to the second hidden state (e.g. j).

to node u', that is a child of node u, if the latter node is associated to the Viterbi hidden state i.

The bi-directional context encoding is obtained by combining the directional context multiset obtained by TD and BU as shown in Fig. 2. In particular, we obtain the unigram and bigram representations of the two models (e.g. through Algorithm 1) and we concatenate them into a $2 \cdot (C + C^2)$-dimensional bi-directional context encoding

$$\Phi^{tb} = [\Phi^{td}_{ug} \Phi^{td}_{bg} \Phi^{bu}_{ug} \Phi^{bu}_{bg}]$$

where the ug and bg subscripts denote unigrams and bigrams, respectively. Once obtained the bi-directional multiset representation, we obtain the tree kernel by exploiting the *Jaccard similarity* [19], [11], that is a well known metric for comparing multisets. In particular, we define the *bi-directional Jaccard kernel* for trees as the multiset Jaccard similarity

$$k_{tb}(\mathbf{x}^1, \mathbf{x}^2) = \frac{\sum_{i=s}^{D} \min(\Phi^{tb}_s(\mathbf{x}^1), \Phi^{tb}_s(\mathbf{x}^2))}{\sum_{s'=1}^{D} \max(\Phi^{tb}_{s'}(\mathbf{x}^1), \Phi^{tb}_{s'}(\mathbf{x}^2))} \qquad (8)$$

where D the feature space size of the multiset bi-directional encodings $\Phi^{tb}(\cdot)$. The Jaccard metric in (8) is a kernel given that it is a similarity function and it is positive definite [20]. Note that the encoding Φ^{tb} is obtained as a concatenation of the BU and TD representations: due to the formulation of the Jaccard kernel in (8), computing the bi-directional kernel is equivalent to summing the TD and BU kernel contributions, i.e. k_{td} and k_{bu} (computed similarly to (8)). The sum formulation is interesting as it allows to generalize the bi-directional kernel to a convex combination of the TD and BU contexts as

$$k_{tb}(\mathbf{x}^1, \mathbf{x}^2) = (1 - \alpha) k_{td}(\mathbf{x}^1, \mathbf{x}^2) + \alpha k_{bu}(\mathbf{x}^1, \mathbf{x}^2),$$

where α is a meta-parameter in $[0, 1]$ that can be used to weight the contribution of one context over the other. Its value can be determined, for instance, as part of a cross-validation process. Alternatively, it can be used to express knowledge on the prior confidence that we have on the contexts. For instance, given acc_{bu} and acc_{td} as the accuracy of the BU and TD kernels on a validation set, the α meta-parameter can be chosen as

$$\alpha = \frac{acc_{bu}}{acc_{bu} + acc_{td}}.$$

Algorithm 1 Viterbi unigram-bigram for a BU model

Require: A topologically sorted tree $\mathbf{x}^n$ with root at$u = 1$ and leaves with indices from U_n (total nodes) back to $I_n + 1$ (I_n is the index of the first non-leaf node).
  for $u = I_n + 1$ **to** U_n **do**
    $\delta_u(i) = \rho_i$ for $i = 1, \ldots, C$
  end for
  for $u = I_n + 1$ **to** 1 **do** *// upwards recursion*
    for $i = 1$ **to** C **do**
      for $l = 1$ **to** L **do**
        $j^* = \arg\max_j \left\{ \varphi_l A^l_{i,j} b_j(x^n_{ch_l(u)}) \delta_{ch_l(u)}(j) \right\}$
        $sum_u = sum_u + \varphi_l A^l_{i,j^*}$
        $prod_u = prod_u \cdot b_{j^*}(x^n_{ch_l(u)}) \delta_{ch_l(u)}(j^*)$
        $ind(i, u, ch_l(u)) = j^*$
      end for
      $\delta_u(i) = sum_u \cdot prod_u$
    end for
  end for
  $Q^*_{n,1} = \arg\max_i \{\delta_u(i) b_i(x^n_1)\}$
  for $u = 1$ **to** I_n **do** *// downwards recursion*
    $i = Q^*_{n,u}$
    $\Phi_i(\mathbf{x}^n) = \Phi_i(\mathbf{x}^n) + 1$ *// unigram*
    for $l = 1$ **to** L **do**
      $j = ind(i, u, ch_l(u))$
      $Q^*_{n,ch_l(u)} = j$
      $\Phi_{i_j}(\mathbf{x}^n) = \Phi_{i_j}(\mathbf{x}^n) + 1$ *// bigram*
    end for
  end for
  return $\Phi^{bu}(\mathbf{x}^n) = [\Phi_i(\mathbf{x}^n) \Phi_{i_j}(\mathbf{x}^n)]$ for $i, j \in [1, \ldots, C]$

III. Experimental Results

The experimental analysis focuses on assessing whether the bi-directional kernel (TB) provides a richer and more effective context with respect to the unidirectional TD and BU kernels. To this end, we have designed an experimental evaluation on real-world challenging benchmarks on tree structured data classification.

The first two benchmarks have been proposed as part of the 2005 and 2006 INEX Competition [21]. The first dataset, referred to as INEX 2005, is the (m-db-s-0) corpus, comprising $9,361$ XML formatted documents represented as trees comprising a total of $124,359$ vertices. Documents are labeled by 11 thematic categories, with consistently varied class distributions, such that node labels represent 1 out of 366 possible XML-tags, modeled by a multinomial distribution. The second dataset, referred to as INEX 2006, is the IEEE corpus, composed of $12,107$ XML formatted documents (for a total of $218,537$ vertices) representing sci-

TABLE I

CLASSIFICATION PERFORMANCE ON THE EXTERNAL (OUT-OF-SAMPLE) TEST SET FOR THE GENERATIVE KERNEL CONFIGURATIONS SELECTED IN CROSS-VALIDATION. PERFORMANCE FOR THE INEX TASK IS EXPRESSED AS CLASSIFICATION ACCURACY (%), WHILE FOR PROPBANK THIS IS EXPRESSED BY F1 SCORE (VARIANCE IS REPORTED IN BRACKETS). THE BEST RESULT FOR EACH DATASET IS HIGHLIGHTED IN BOLD.

	BU		TD		TB	
Dataset	Size	Test	Size	Test	Size	Test
INEX05 (accuracy)	$C = 8$	94.22 (0.81)	$C = 10$	93.39 (2.19)	$C = 8$	**95.39** (0.14)
INEX06 (accuracy)	$C = 6$	44.53 (0.09)	$C = 8$	44.38 (0.06)	$C = 10$	**44.78** (0.02)
Propbank (F1 Score)	$C = 8$	0.567 ($< 10^{-3}$)	$C = 10$	0.577 ($< 10^{-3}$)	$C = 10$	**0.586** ($< 10^{-3}$)

entific articles, each from one of 18 different IEEE journals. Node emission has been modeled by a multinomial whose elements represent 1 out of 65 different XML tags. The large number of classes in both data sets makes them challenging benchmarks, such that the random classifier baseline for INEX 2005 and INEX 2006 is 9% and 5.5%, respectively. Both datasets provide standard splits for training and test samples [22], such that INEX 2005 comprises 4820 training structures and $4,811$ test samples, while INEX 2006 is split into $6,053$ training trees and $6,054$ test data.

The third benchmark, i.e. Propbank, deals with the classification of parse trees representing English propositions and associated semantic information. The Propbank dataset derives from the Penn English TreeBank and consists of material from a set of Dow-Jones news articles, divided into sections. For the purpose of this paper, we use a sample of trees from section 24, including $7,000$ training trees and $2,000$ validation examples, as well as $6,000$ test samples extracted from section 23 [13]. The trees in the dataset have outdegree 15 and an average number of nodes equal to 13.95. The label alphabet is huge and includes $6,654$ elements corresponding to English words and semantic annotations. This benchmark defines a binary classification problem with a very unbalanced class distribution, where the percentage of positive examples in each set is roughly 7%. For this reason, the F1 score is used to measure the predictive accuracy of the kernels. The dataset comes with a standard split into training, validation and test sets.

Different configurations of the TD and BU generative models have been tested by varying the number of hidden states C in $\{6, 8, 10\}$. The number of tested hidden states has been determined following the guidelines in [6]. Note that both TD and BU train a different model for each class, hence the total number of hidden states for a task is $C \cdot V$, where V is the number of classes. Training and test trials have been repeated 5 times for each configuration of the generative models, each time using different random initializations for the models distributions. Only the initialization of the emission distribution is kept fixed, by using the prior distribution of the multinomial node labels estimated solely on the training trees.

SVM-based multiclass classification has been obtained by means of the LIBSVM[1] software (by a C-SVM classifier) exploiting the Jaccard Gram matrices as user defined kernel.

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm

TABLE II

TEST CLASSIFICATION ACCURACY ON THE INEX 2005 DATA AS A FUNCTION OF THE HIDDEN STATE SIZE. THE CONFIGURATION SELECTED IN CROSS-VALIDATION IS HIGHLIGHTED IN BOLD.

Kernel	$C = 6$	$C = 8$	$C = 10$
BU	91.56 (0.81)	**94.22** (0.81)	93.81 (0.81)
TD	89.71 (1.57)	93.04 (2.93)	**93.39** (2.19)
TB	92.45 (0.56)	**95.39** (0.14)	94.76 (0.27)

A cross-validation (CV) procedure has been applied to select the value of the misclassification cost parameter C_{svm} from the following set of values: 0.001, 0.01, 0.1, 1, 10, 100, 1000 using validation data external to the test set. In particular, for the INEX tasks we have applied a 3-fold CV to the training set (using classification error), while Propbank comes with a standard validation set which has been used to perform model selection on the C_{svm} misclassification cost (assessing validation performance with the F1 score). The value used for α is $\frac{1}{2}$: this choice is likely to return an underestimation of the performance that can be obtained by optimizing α in CV, but we are interested in assessing the contribution of having a bi-directional context rather than seeking an optimized kernel. The SVM classifier with the C_{svm} value selected on the validation set has then been trained and its average performance on the 5 trials has been evaluated on the hold-out test data.

Table I shows the test classification performance on the three datasets for the model-selected configurations of the generative kernels (and associated probabilistic models). The results show that TB achieves a classification performance that is significantly higher (given the low variance) than that achieved by the TD and BU unidirectional approaches, suggesting that it is effectively combining the context captured by the two underlying generative models. This behavior is more evident when considering the detailed results for the INEX 2005 dataset reported in Table II: here, we compare the performance of the three approaches for the same hidden state size C. Note that, for a given C, the TB result is obtained from the same bottom-up and top-down generative models used to obtain the BU and TD results, respectively. These are only combined in 5 random couples to yield to 5 TB trials per each configuration of C. In other words, the bi-directional context effectively combines the information captured in the generative models with respect to an uni-directional context applied to the very same generative models.

By taking a closer look at the results in Table I, it is interesting to note that certain tasks are better suited to a bottom-up approach, such as INEX 2005 and INEX 2006, while Propbank obtains a better result when a top-down context is used. This aspect points out another significant advantage of the TB approach, as it appears to be capable of getting better generalization performance across a variety of learning task with respect to the unidirectional approaches. When compared to other results in literature, the TB kernel shows a competitive performance, in particular as regards the INEX 2006 and Propbank data. The former dataset is an hard benchmark, generally yielding to low classification performances: [23] provides a recent survey of the results of several tree classification approaches on the INEX 2006 dataset. The results in [23] show that TB achieves the best classification accuracy in literature on the INEX 2006 dataset: state-of-the-art syntactic kernels such as ST [24] and SST [25] yield to test classification accuracies of 32.02% and 40.41%, respectively. Similarly, in [13], it is provided an aggregated view of the results of the main syntactic tree kernels in literature on the Propbank dataset. These results show that TB achieves again the best F1 score: e.g. ST and SST yield to 0.517 and 0.542, respectively. TB also outperforms an enhanced version of the PT kernel which allows richer contexts enabling partial matches between subtrees and that scores an F1 measure of 0.579 [13].

IV. Conclusions

We have introduced an approach that easily and effectively combines the structural information captured by generative tree models characterized by top-down and bottom-up contexts. The proposed approach exploits the idea of hidden states multisets to realize a tree encoding that takes into account both the summarized information on the path leading to a node (i.e. a top-down context) as well as the information on how substructures are composed to create a subtree rooted on a node (bottom-up context). We have highlighted how different contexts yield to different probabilistic assumptions in the generative models and, more practically, to different performances depending on the learning task the are applied to. The experimental analysis shows that by taking a bi-directional approach incorporating both the top-down and bottom-up contexts yields to superior performances with respect to the unidirectional approach. More importantly, the proposed TB kernel achieves state of the art results on challenging tree classification benchmarks, i.e. INEX 2006 and Propbank. The method proposed in the paper to combine different contextual information is simple, but paves the way to more articulated approaches exploiting the same underlying intuition. In Section II-B, we have already suggested more refined ways of combining independent bottom-up and top-down multisets. Nevertheless, we believe that a considerable advancement can be obtained through the introduction of *hybrid* encodings, such that the state-gram encodes information on the occurrence of TD and BU hidden states in a parent-child relationships, allowing the kernel to capture more discriminative structural matches.

References

[1] P. Baldi, S. Brunak, P. Frasconi, G. Pollastri, and G. Soda, "Bidirectional dynamics for protein secondary structure prediction," in *Sequence Learning*. Springer, 2001, pp. 80–104.

[2] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi, "Tree automata techniques and applications," 2007.

[3] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Readings in Speech Recognition*, pp. 267–296, 1990.

[4] S. Lauritzen, *Graphical models*. Oxford University Press, USA, 1996.

[5] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Trans. Neural Networks*, vol. 9, no. 5, pp. 768–786, 1998.

[6] D. Bacciu, A. Micheli, and A. Sperduti, "Compositional generative mapping for tree-structured data - part I: Bottom-up probabilistic modeling of trees," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 23, no. 12, pp. 1987–2002, 2012.

[7] M. Crouse, R. Nowak, and R. Baraniuk, "Wavelet-based statistical signal-processing using hidden markov-models," *IEEE Trans. Signal Process.*, vol. 46, no. 4, pp. 886–902, April 1998.

[8] M. Diligenti, P. Frasconi, and M. Gori, "Hidden tree markov models for document image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 4, pp. 519–523, 2003.

[9] B. Hammer, A. Micheli, and A. Sperduti, "Universal approximation capability of cascade correlation for structures," *Neural Computation*, vol. 17, no. 5, pp. 1109–1159, 2005.

[10] A. Micheli, D. Sona, and A. Sperduti, "Contextual processing of structured data by recursive cascade correlation," *IEEE Transactions on Neural Networks*, vol. 15, no. 6, pp. 1396–1410, 2004.

[11] D. Bacciu, A. Micheli, and A. Sperduti, "A generative multiset kernel for structured data," in *Proc. of ICANN'12*, ser. LNCS, vol. 7552. Springer, 2012, pp. 57–64.

[12] F. Aiolli, G. Da San Martino, and A. Sperduti, "Route kernels for trees," in *Proc. of ICML '09*. ACM, 2009, pp. 17–24.

[13] ——, "Extending tree kernels with topological information," in *Proc. of ICANN'11*, ser. LNCS. Springer, 2011, pp. 142–149.

[14] ——, "A new tree kernel based on SOM-SD," in *Artificial Neural Networks–ICANN 2010*, ser. LNCS, vol. 6353. Springer, 2010, pp. 49–58.

[15] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.

[16] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[17] J. Durand, P. Goncalves, Y. Guedon, I. Rhone-Alpes, and F. Montbonnot, "Computational methods for hidden Markov tree models-an application to wavelet trees," *IEEE Trans. Signal Process.*, vol. 52, no. 9, pp. 2551–2560, 2004.

[18] D. Bacciu, A. Micheli, and A. Sperduti, "An input-output hidden Markov model for tree transductions," *Neurocomputing*, vol. 112, pp. 34–46, 2013.

[19] P. Jaccard, "The distribution of the flora in the alpine zone.1," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.

[20] J. C. Gower, "A general coefficient of similarity and some of its properties," *Biometrics*, pp. 857–871, 1971.

[21] L. Denoyer and P. Gallinari, "Report on the XML mining track at INEX 2005 and INEX 2006: categorization and clustering of XML documents," *SIGIR Forum*, vol. 41, no. 1, pp. 79–90, 2007.

[22] F. Aiolli, G. Da San Martino, M. Hagenbuchner, and A. Sperduti, "Learning nonsparse kernels by self-organizing maps for structured data," *IEEE Trans. on Neural Netw.*, vol. 20, no. 12, pp. 1938–1949, 2009.

[23] C. Gallicchio and A. Micheli, "Tree echo state networks," *Neurocomputing*, vol. 101, pp. 319–337, 2013.

[24] S. V. N. Vishwanathan and A. J. Smola, "Fast kernels for string and tree matching," in *Advances in Neural Information Processing Systems 15*, 2003, pp. 569–576.

[25] M. Collins and N. Duffy, "New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron," in *Proc. of ACL 2002*, 2002, pp. 263–270.