View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Institutional Research Information System University of Turin

Building a wide coverage dynamic grammar

Alessandro Mazzei and Vincenzo Lombardo

Dipartimento di Informatica, Università di Torino
c.so Svizzera, 185 10149, Torino, Italy
[mazzei|vincenzo]@di.unito.it

Abstract. Incremental processing is relevant for language modeling, speech recognition and language generation. In this paper we devise a dynamic version of Tree Adjoining Grammar (DVTAG) that encodes a strong notion of incrementality directly into the operations of the formal system. After discussing the basic features of DVTAG, we address the issue of building of a wide coverage grammar and present novel data on English and Italian.

1 Introduction

Incrementality is a largely held assumption that constrains the language processor to parse the input words from left to right, and to carry out a semantic interpretation of the partial structures [1]. Most of the approaches to incremental parsing assume some standard syntactic formalism (e.g. context-free grammar) and address the issue of incrementality in the parsing algorithm [2]. An alternative approach (*incrementality in the competence*) is to encode incrementality in the operations of a formal system. Competence and some performance issues are addressed by the same model, derivation and parsing share the same mechanism, and we can use the two terms interchangeably (we can generically talk of a *syntactic process*, cf. [3]).
The detailed specification of the incremental syntactic process is often addressed by assuming a parsimonious version of incrementality that we can call *strong connectivity* [4]. Strong connectivity constrains the syntactic processor to maintain a fully connected structure throughout the whole process and is supported by a large amount of psycholinguistic evidence [5, 6] as well as linguistic facts [7]. The syntactic process consists in attaching each word from left to right to the existing unique structure, called the *left-context*, that spans the previous words in the string.
Tree Adjoining Grammars (TAG) is a well known family of formalisms. Recently the results of some psycholinguistic experiments have suggested that the adjoining mechanism of TAG can be used to fulfill the to the strong connectivity hypothesis (*adjoining mechanism hypothesis*) [6].
A natural way of viewing the syntactic process is a dynamic system, that is a system that evolves in time through a number of steps. A *dynamic grammar* views the syntactic process as a sequence of transitions between adjacent states S_{i-1}

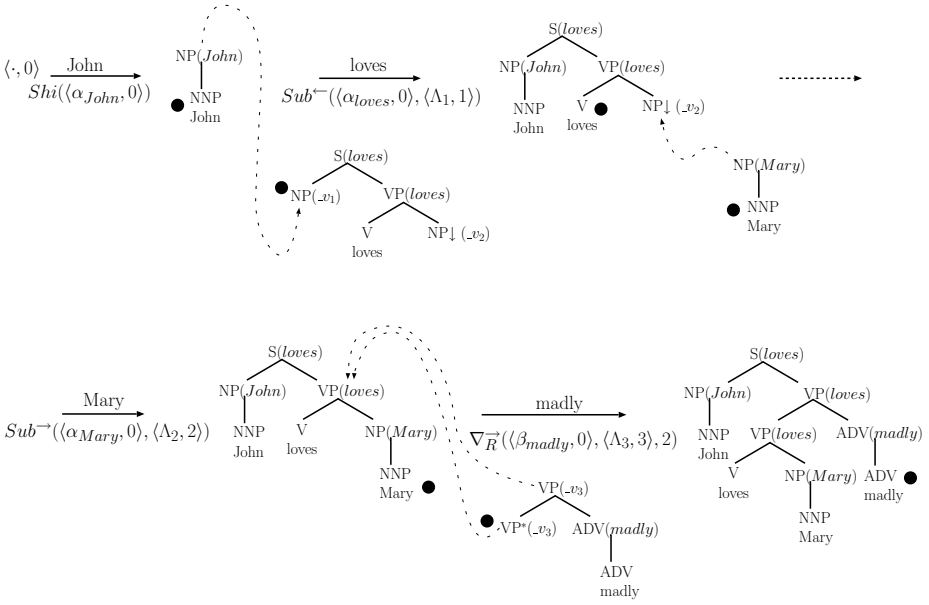

Fig. 1. The DVTAG derivation of the sentence *John loves Mary madly.*

and S_i while moving from left to right on the string of terminals [8]. Thus, it naturally fulfills the incrementality in the competence and the strong connectivity hypotheses. In this paper, we describe a constituency based dynamic grammar, called *Dynamic Version of TAG* (DVTAG), that fulfills the hypotheses of *incrementality in the competence*, *strong connectivity*, *adjoining mechanism*. DVTAG is TAG-related, since it shares some basic properties with LTAG formalism, i.e. the extended domain of locality, the lexicalization and the adjoining mechanism [9].
The focus of the paper is on the applicability of the formalism into a realistic context. In particular, we build two wide coverage DVTAGs for English and Italian respectively, through extraction from treebank.

2 LTAG, Dynamic grammars and DVTAG

In Fig. 1 we can see the DVTAG derivation of the sentence *John loves Mary madly.* Like LTAG [10], a Dynamic Version of Tree Adjoining Grammar (DVTAG) consists of a set of elementary trees, divided into initial trees and auxiliary trees, and attachment operations for combining them. Lexicalization is expressed through the association of a lexical *anchor* with each elementary tree. With the aim to consider the lexical dependencies between the lexical items, each node in the elementary trees is augmented with a feature indicating the lexical head

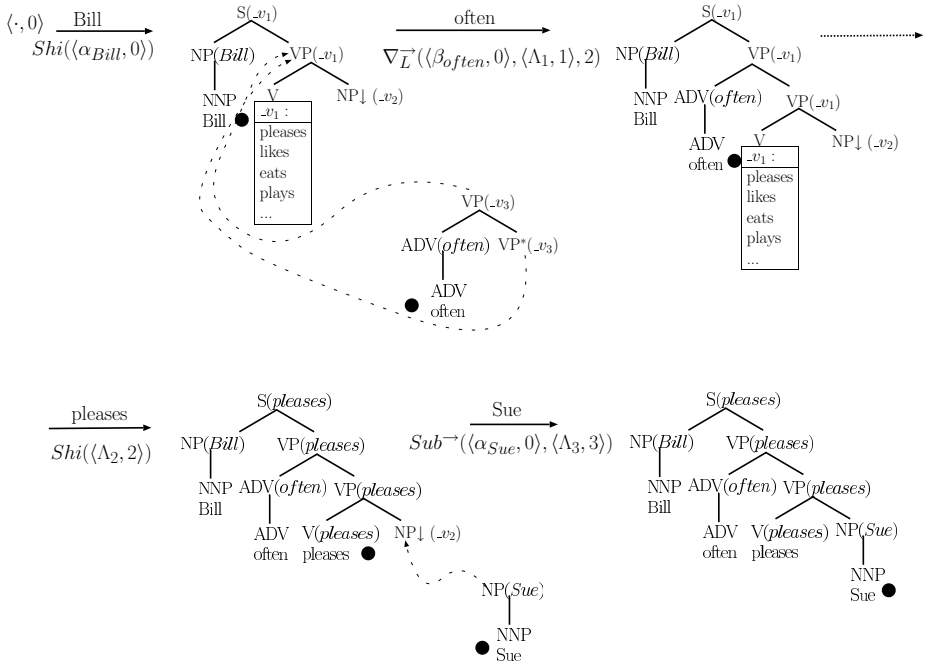

Fig. 2. The DVTAG derivation of the sentence *Bill often pleases Sue*.

that projects the node. The head variable is a variable in logic terms: $\_v_3$ will be unified with the constant *loves* in the derivation of Fig. 1. The derivation process in DVTAG builds a constituency tree by combining the elementary trees via some operations that are illustrated below. DVTAG implements the incremental process by constraining the *derivation process* to be a series of steps in which an elementary tree is combined with the partial tree spanning the left fragment of the sentence. The result of a step is an updated partial structure. Specifically, at the processing step i, the elementary tree anchored by the i-th word in the sentence is combined with the partial structure spanning the words from 1 to $i-1$ positions; the result is a partial structure spanning the words from 1 to i. In DVTAG the derivation process starts from an elementary tree anchored by the first word in the sentence and that does not require any attachment that would introduce lexical material on the left of the anchor (such as in the case that a Substitution node is on the left of the anchor). This elementary tree becomes the first left-context that has to be combined with some elementary tree on the right. At the end of the derivation process the left-context spans the whole sentence, and is called the *derived tree*: the last tree of Fig.1 is the derived tree for the sentence *John loves Mary madly*.
In DVTAG we always combine a left context with an elementary tree, then

there are seven attachment operations. Standard LTAG adjoining is split into two operations: *adjoining from the left* and *adjoining from the right*. The type of adjoining depends on the position of the lexical material introduced by the auxiliary tree with respect to the material currently dominated by the adjoined node (which is in the left-context). In Fig. 1 we have an adjoining from the right in the case of the left auxiliary tree anchored by *madly*, and in Fig. 2 we have an adjoining from the left in the case of the left auxiliary tree anchored by *often*. Inverse operations account for the insertion of the left-context into the elementary tree. In the case of *inverse substitution* the left-context replaces a substitution node in the elementary tree; in the case of *inverse adjoining from the left* and *inverse adjoining from the right*, the left-context acts like an auxiliary tree, and the elementary tree is split because of the adjoining of the left context at some node[1]. In Fig. 1 we have an inverse substitution in the case of the initial tree anchored by *John*. Finally, the *shift* operation either scans a lexical item which has been already introduced in the structure or derives a lexical item from some predicted preterminal node. The grounding of the variable $\_v_1$ in Fig. 2 is an example of shift.

It is important to notice that, during the derivation process, not all the nodes in the left-context and the elementary tree are accessible for performing operations: given the $i-1$-th word in the sentence we can compute a set of accessible nodes in the left-context (the *left-context fringe*); also, given the lexical anchor of the elementary tree, that in the derivation process matches the i-th word in the sentence, we can compute a set of accessible nodes in the elementary tree (the *elementary tree fringe*). To take into account of this feature, the elementary tree in DVTAG are dotted tree, i.e. a couple $\langle \gamma, i \rangle$ formed by a tree γ and an integer i denoting the accessible fringe[2] of the tree [11].

The DVTAG derivation process requires the full connectivity of the left-context at all times. The extended domain of locality provided by LTAG elementary trees appears to be a desirable feature for implementing full connectivity. However, each new word in a string has to be connected with the preceding left-context, and there is no *a priori* limit on the amount of structure that may intervene between that word and the preceding context. For example, in a DVTAG derivation of *John said that tasty apples were on sale*, the adjective *tasty* cannot be directly connected with the S node introduced by *that*; there is an intervening NP symbol that has not yet been predicted in the structure. Another example is the case of an intervening modifier between an argument and its predicative head, like in the example *Bill often pleases Sue* (see Fig.2). The elementary tree *Bill* is linguistically motivated up to the NP projection; the rest of the structure depends on connectivity. These extra nodes are called *predicted nodes*. A predicted preterminal node is referred by a set of lexical items, that represent a *predicted*

[1] In [9] there is shown the importance of the inverse operation to obtain the correct cross-serial dependencies in DVTAG.

[2] In the picture we represent the integer using a dot. Note that *fringe* and *dotted tree* are two concepts borrowed from parsing as a consequence of the dynamic nature of DVTAG.

head. So, the extended domain of locality available in LTAG has to be further extended. In particular, some structures have to be predicted as soon as there is some evidence from arguments or modifiers on the left.
The notion of predicted nodes is crucial for the linguistic appealing of the formalism. Indeed DVTAG does not assume any linguistic theory *a priori*. Similar to LTAG, DVTAG allows for the specification of a number of linguistic principles licensing the elementary trees. The mathematical properties of TAG are a consequence of the finiteness of the size of the lexicon, then the linguistic principles have to guarantee (more or less implicitly) this property [12]. In DVTAG there is no *a priori* limit on the number of predicted nodes, an then there is not a limit on the size of the grammar. The notion of predicted nodes is also crucial for the applicability of DVTAG a real context. Several works have showed that the most important factor in TAG parsing complexity is the size of the grammar [13] [14], and for a wide coverage DVTAG predicted nodes could produce a very huge grammar [15]. Our hypothesis is that DVTAG respects the constraint on the finiteness of the lexicon because there is an empirical limit on the number of predicted heads. This limit can be obtained by the observation of the experimental data: several work confirmed that the number of predicted heads necessary to guarantee the strong connectivity is relatively low [16, 17].
In the next sections we address the problem of building a linguistic motivated wide coverage DVTAG for English and Italian. As a side result of these experiments we present a number of exploratory experiments that empirically verify the hypothesis about the limited number of predicted heads.

3 Building a wide coverage DVTAG

A way to build a wide coverage grammar is to manually write the grammar. For instance, XTAG [18] is an ongoing project to produce a hand written wide coverage LTAG for English. In [15] and [11] we have discussed the possibility of transforming the XTAG grammar into a DVTAG using a non-constrained transformation. These experiments have showed the DVTAG produced in this way is not adequate for an applicative context. The key point is the size of the grammar. Indeed the non-constrained transformation did not take into account any empirical principle to construct the DVTAG elementary trees. In this paper we pursue a different approach.
As large size treebanks are becoming widely available (among others Penn treebank [19]) a new way is the automatic extraction of wide coverage grammar from a treebank [20, 21]. Here we investigate on the possibility of extracting a large DVTAG from a treebank. In particular, exploiting the formal relation between LTAG and DVTAG (cf. [11]) we propose a two steps algorithm. In the first step we use a LTAG extractor; in the second step we transform the LTAG produced in the first step into a DVTAG. To pursue this strategy we use the LTAG extractors described in [20] and [14] for English and Italian respectively. Both these extractors have been used to test the relation between the time complexity of the parsing and the size of LTAG lexicon [13, 14]. The Italian LTAG extractor

has been also used to test the relation between the coverage of extracted LTAG and the genre of the extraction corpus [22].
In section 3.1 we briefly describe the basic issues of an algorithm to automatically extract a LTAG from a treebank. In section 3.2 we define the *left-association*, an operation that allows us to increase a DVTAG lexicon with new wider elementary trees. In section 3.3 we describe an algorithm to transform a LTAG automatically extracted into a DVTAG based on the left-association. In section 3.4 and in section 3.5 we report some data for DVTAG automatically extracted from an English treebank and from an Italian treebank respectively: we provide an explorative analysis of this data with the aim to test the reliability of the grammars produced.

3.1 Extracting a LTAG from a treebank

All the algorithms for automatic extraction of a LTAG from a treebank share a basic feature. They produce a LTAG grammar that covers the treebank, and at the same time they assign a derivation tree D_T to each tree T of the treebank ([20] [21]). Each derivation tree D_T contains all the elementary trees that the algorithm extracts from each derived tree T. The derivation trees D_T play a key role in the definition of a probabilistic model for the treebank grammar [23]. Now we describe the basic feature of the algorithm (henceforth LexTract) proposed in [20] for the extraction of a wide coverage LTAG from Penn treebank. A quite similar algorithm has been used for the extraction of a wide coverage LTAG from an Italian dependency treebank [14]. LexTract can be essentially described as a two steps procedure. In the first step, each constituency tree T of the treebank is converted into a set of elementary trees $\gamma_1^T ... \gamma_n^T$. By using a head-table and a modifier-table, the algorithm identifies the head and modification relations between the nodes of constituency tree. The algorithm uses the head-modifier annotation together with a set of prototypical tree templates (a sort of extended projection skeletons) to extract a set of elementary trees that derives the constituency tree. In the second step, using the elementary trees $\gamma_1^T ... \gamma_n^T$ produced in the first step, each constituency tree T of the treebank is converted into a derivation tree.

3.2 Left-association

The left-association takes as input two DVTAG elementary trees, called the **base tree** and the **raising tree** respectively, and returns a new DVTAG elementary tree, the **raised tree**[3]. The operation produces the raised tree by grafting the base tree into the raising tree, and replacing the left-anchor of the raising tree with a new head-variable. Left-association can be performed during the parsing/derivation process (i.e. on-line, cf. [17]) or with the goal to extend

[3] There are some similarities between left-association and the CCG type raising operation [3], because in both cases some root category X is raised to some higher category Y.

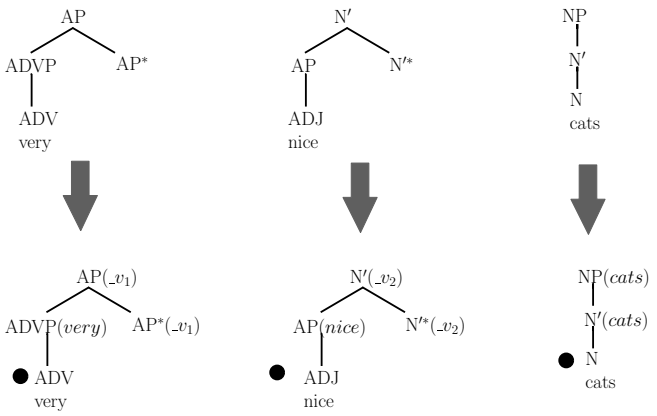

Fig. 3. First step of the conversion from a LTAG to a DVTAG: each nodes is augmented with a head variable.

the lexicon (i.e. off-line). In this paper we are exploring the consequences of increasing the role of the competence grammar, then we use this operation off-line. The raised tree produced by the left-association displays a larger number of predicted nodes in comparison with the base tree. The raised tree has zero or more predicted heads: the raised tree has one more predicted head that projects a number of the predicted nodes in the raised tree (cf. Fig. 4).

3.3 Converting an automatically extracted LTAG into a DVTAG

The left-association can be used to transform an automatically extracted LTAG into a DVTAG. For each tree T in the treebank, we define a relation, called *left-corner* relation, on the nodes belonging to D_T. We say that γ_k γ_l are in the left-corner relation if in D_T γ_k is the left-most child of γ_l, and if the root node label of γ_k is different from the root node label of γ_l. If two nodes γ_k γ_l of D_T are in the left-corner relation, we use the left-association on the respective trees to build a new DVTAG elementary tree.

Conversion algorithm
[step 1:] We extract the LTAG $\gamma_1^T ... \gamma_n^T$ and the derivation D_T from T (i.e. $derived(D_T) = T$). We build the dotted trees $\langle \gamma_1^T, 0\rangle ... \langle \gamma_n^T, 0\rangle$ augmenting each non terminal node in γ_i $i \in 1...n$ with a head feature that contains the lexical item projecting the node (Fig. 3).
[step 2:] To produce $\langle \delta_1, 0\rangle ... \langle \delta_m, 0\rangle$ we apply the transitive closure of the left-association on $\langle \gamma_1, 0\rangle ... \langle \gamma_n, 0\rangle$.

The application of the transitive closure of left-association is subject to a **closure condition**. The sequence of raising tree in the transitive closure of the left-association, applied to some base tree $\langle \delta, 0\rangle$, has to respect the left-corner relation.

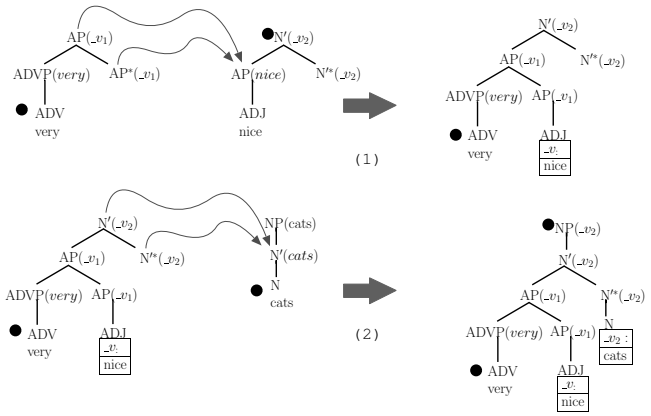

Fig. 4. Second step of the conversion from a LTAG to a DVTAG: iteration of the left-association.

The adequacy of the closure condition is motivated by the presence of the adjoining operation, which factorizes recursion. So, adjoining should account for repetitions of the root category outside of the left-association. But, since DVTAG respects the strong connectivity hypothesis, not all the cases of recursion in LTAG derivation tree are generable in DVTAG also using the predicted nodes (cf. [11]). Then, in some cases the generative power of the DVTAG produced with the conversion algorithm is not equal to the generative power of the original LTAG extracted. However, our final goal in this work is to provide an exploratory analysis of a realistic DVTAG. In comparison with this task, we assume as a working hypothesis that the difference on the generative power between LTAG and DVTAG is not substantially relevant.
In the next sections we apply the conversion procedure described above to two LTAG automatically extracted from an English and Italian treebanks respectively. We compare the properties of the DVTAG extracted with the data reported in [16], based on *Connection Path Grammars* (CPG). Similar to DVTAG, CPG is a dynamic constituency grammatical formalism, but there are some crucial differences between them. DVTAG uses linguistically motivated elementary trees and uses adjoining to factorize recursion. In contrast, CPG uses elementary structures motivated uniquely by the constraint of strong incrementality. However, similarly to DVTAG CPG includes the notion of predicted heads, and then can be used as a baseline.

3.4 Extraction of an English DVTAG from Penn Treebank

By applying the LexTract[4] on the sections 02-21 of the Wall Street Journal part of the Penn treebank II (39,832 sentences, 950,026 words), we have obtained

[4] We wish to thank Fei Xia, that kindly let us to use her program.

# of iterations of left-association	# DVPenn dotted trees	% DVPenn dotted trees	% Connection paths
0	764,359	90.95	82,33
1	75,125	8.94	15,26
2	959	0.11	2,28
3	7	0.00	0.13
Total:	840,450		

Table 1. Number of (non distinct) DVTAG elementary dotted trees with respect to the number of left-associations for the DVPenn grammar.

# of iteration of left-association	# DVPenn dotted templates	% DVPenn dotted templates	# Connection paths
0	4,947	41.13	
1	6,329	52.63	
2	743	6.18	
3	7	0.06	
Total:	12,026		1,896

Table 2. Number of DVTAG templates with respect to the number of left-associations for the DVPenn grammar.

841,316 LTAG (non distinct) elementary trees[5], corresponding to 5,268 (distinct) templates. The DVTAG conversion algorithm on these LTAG elementary trees, produces a DVTAG (henceforth DVPenn) with 840,450 (non distinct) elementary dotted trees corresponding to 12,026 (distinct) dotted templates. In Table 1 we have reported the number of dotted trees with respect to the number of left-association in DVPenn. Moreover we have reported the also the percentage of CPGs with respect to the number of *headless* as reported in [16]. Note that we have used the percentage because in in [16] it has been used a fraction of the WSJ of ∼100,000 words: then their corpus is ten times smaller with respect to corpus used in this experiment. Comparing the percentage of dotted trees in DVPenn to the percentage of connection path in the CPG extracted from Penn, we note that both the grammars have really few elementary structures with more than two predicted heads (∼0.001% and 0.13% for DVPenn and CPG respectively). Moreover, most of the structures do not use predicted heads (90,95% and 82,33% for DVPenn and CPG respectively). In Table 2 we have reported the number of dotted templates with respect to the number of left-association

[5] The number of elementary tree is different in comparison with the number of token words because LexTract does not extract trees anchored by punctuations.

# of iterations of left-association	# DVTUT dotted trees	% DVTUT dotted trees	% Connection paths
0	37,382	92.39	82,33
1	3,059	7.56	15,26
2	22	0.05	2,28
3	0	0.00	0.13
Total:	40,463		

Table 3. Number of DVTAG trees with respect to the number of left-association, for the DVTUT grammar.

in DVPenn[6]. In this case, the maximum percentage of dotted templated (52,63 %) are left-associated only once. Also if the corpus used in [16] is smaller with respect to our corpus, the total number of templates extracted is not really different for CPG, LTAG and DVTAG, 1,896 5,268 12,026 respectively. This fact could suggest that the number of structures necessary to fulfill strong connectivity is not really huge, but more experiments on the properties of the extracted grammars are necessary. For example to assess the quality of the grammars it is necessary to test their *coverage*.
However the results of the experiment show that for the Penn treebank only few dotted trees have more than two predicted heads. Moreover, this result is in accord with a precedent test performed with connection path grammars [16]. In the next section we replicate this experiment using an Italian treebank.

3.5 Extraction of an Italian DVTAG from TUT

The Turin University Treebank (TUT) is an ongoing project of the University of Turin on the construction of a dependency style treebank for Italian [24]: each sentence is semi-automatically annotated with dependency relations that form a tree, and relations are of morphological, syntactic and semantic types. The corpus is very varied, and contains texts from newspapers, magazines, novels and press news. Its current size is 1,800 annotated sentences (40,470 words). In order to extract the LTAG grammar, we have converted the TUT treebank dependency format to a constituency format[7], and then we have adapted the LexTract algorithm described above (for the details of the algorithms see [14]).
We have used the DVTAG extraction algorithm by applying the TUT LTAG extractor on the whole TUT corpus. From the 167 out 1800 sentences are discarded because they represent linguistic constructions that the extractor is not yet able to take into account. From the remanent 1683 sentences, the LTAG extractor produces (non distinct) $43,621$ elementary trees corresponding to $1,212$

[6] In this measure we do not have the complete data for CPG.
[7] The constituency format is not yet compliant with the Penn treebank format

# of iterations of left-association	# DVTUT dotted templates	% DVTUT dotted templates	# Connection Path templates
0	844	52.29	
1	752	46.59	
2	18	1.12	
Total:	1,614		1,896

Table 4. Number of DVTAG templates with respect to the number of left-association, for the DVTUT grammar.

(distinct) templates, while the DVTAG extractor produces 40,463 dotted trees corresponding to 1,614 dotted templates (henceforth DVTUT grammar).
In Table 4, we have reported the number of templates with respect to the number of left-association. The total number of dotted templates is 1,614: 844 are no left-associated, 752 are left-associated only once, 18 are left-associated two times. The percentage values for zero or one left-associations is quite close (52% vs. 46.59%), while the number of two times left-associated templates is very small (1,12%). In the last raw of the table, we have compared the total number of dotted templates with the total number of *connection path type* (i.e. template) reported in [16]. Note that also if the size of the TUT is less than the half of the corpus used in [16] (40,470 versus ∼100,000 words respectively), the number of templates is quite similar. However as well as for the experiment on the Penn treebank, the results of this experiment are in accord with the results of [16].

4 Conclusions

In this paper we have provided the basic features of the DVTAG formalism. We have analyzed the theoretical and applicative problems related to the notion of predicted heads. Moreover, we have defined an algorithm to extract a DVTAG from a treebank based on a generic LTAG extractor. We have used this algorithm to produce two wide coverage DVTAGs for English and Italian. Analyzing these grammars, as a side result we have found that the DVTAG elementary trees extracted from the treebanks have very few predicted heads. In a future work we want to extend the analysis of the extracted grammars taking into account of their coverage [22].

References

1. Marslen-Wilson, W.: Linguistic structure and speech shadowing at very short latencies. Nature **244** (1973) 522–523
2. Abney, S.P., Johnson, M.: Memory requirements and local ambiguities of parsing strategies. Journal of Psycholinguistic Research **20** (1991) 233–250

3. Steedman, M.J.: The syntactic process. A Bradford Book, The MIT Press (2000)
4. Stabler, E.P.: The finite connectivity of linguistic structure. In Clifton, C., Frazier, L., Reyner, K., eds.: Perspectives on Sentence Processing. Lawrence Erlbaum Associates (1994) 303–336
5. Kamide, Y., Altmann, G.T.M., Haywood, S.L.: The time-course of prediction in incremental sentence processing: Evidence from anticipatory eye movements. Journal of Memory and Language Language **49** (2003) 133–156
6. Sturt, P., Lombardo, V.: Processing coordinated structures: Incrementality and connectedness. Cognitive Science **29** (2005) 291–305
7. Phillips, C.: Linear order and constituency. Linguistic Inquiry **34** (2003) 37–90
8. Milward, D.: Dynamic dependency grammar. Linguistics and Philosophy **17** (1994)
9. Lombardo, V., Sturt, P.: Towards a dynamic version of TAG. In: TAG+6. (2002) 30–39
10. Joshi, A., Schabes, Y.: Tree-adjoining grammars. In Rozenberg, G., Salomaa, A., eds.: Handbook of Formal Languages. Springer (1997) 69–123
11. Mazzei, A.: Formal and empirical issues of applying dynamics to Tree Adjoining Grammars. PhD thesis, Dipartimento di Informatica, Università degli studi di Torino (2005)
12. Frank, R.: Phrase Structure Composition and Syntactic dependencies. The MIT Press (2002)
13. Sarkar, A., Xia, F., Joshi, A.: Some experiments on indicators of parsing complexity for lexicalized grammars. In: COLING00. (2000)
14. Mazzei, A., Lombardo, V.: Building a large grammar for italian. In: LREC04, Lisbon (2004) 1–4
15. Lombardo, V., Mazzei, A., Sturt, P.: Competence and performance grammar in incremental parsing. In: "Incremental Parsing: Bringing Engineering and Cognition Together", Workshop at ACL-2004, Barcelona (2004) 1–8
16. Lombardo, V., Sturt, P.: Incrementality and lexicalism: A treebank study. In Stevenson, S., Merlo, P., eds.: Lexical Representations in Sentence Processing. John Benjamins (2002)
17. Schneider, D.: Parsing and incrementality. PhD thesis, University of Delaware, Newark (1998)
18. Doran, C., Hockey, B., Sarkar, A., Srinivas, B., Xia, F.: Evolution of the xtag system. In Abeillé, A., Rambow, O., eds.: Tree Adjoining Grammars. Chicago Press (2000) 371–405
19. Marcus, M., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of english: The penn treebank. Computational Linguistics **19** (1993) 313–330
20. Xia, F.: Automatic Grammar Generation from two Different Perspectives. PhD thesis, Computer and Information Science Department, Pensylvania University (2001)
21. Chen, J.: Towards Efficient Statistical Parsing using Lexicalized Grammatical Information. PhD thesis, Computer and Information Science Department, University of Delaware (2001)
22. Mazzei, A., Lombardo, V.: A comparative analysis of extracted grammars. In: 16th European Conference on Artificial Intelligence, ECAI04, Valencia (2004) 1–5
23. Joshi, A., Sarkar, A.: Tree adjoining grammars and their apllication to statistical parsing. In Bod, R., Scha, R., Sima'an, K., eds.: Data-Oriented Parsing. CSLI Publications (2003) 255–283
24. Bosco, C., Lombardo, V., Vassallo, D., Lesmo, L.: Building a treebank for italian: a data-driven annotation schema. In: LREC00, Athens (2000) 99–105