A probabilistic generative model for an intermediate constituency-dependency representation

Federico Sangati
Institute for Logic, Language and Computation
University of Amsterdam, the Netherlands
f.sangati@uva.nl

Abstract

We present a probabilistic model extension to the Tesnière Dependency Structure (TDS) framework formulated in (Sangati and Mazza, 2009). This representation incorporates aspects from both constituency and dependency theory. In addition, it makes use of *junction* structures to handle coordination constructions. We test our model on parsing the English Penn WSJ treebank using a re-ranking framework. This technique allows us to efficiently test our model without needing a specialized parser, and to use the standard evaluation metric on the original Phrase Structure version of the treebank. We obtain encouraging results: we achieve a small improvement over state-of-the-art results when re-ranking a small number of candidate structures, on all the evaluation metrics except for chunking.

1 Introduction

Since its origin, computational linguistics has been dominated by Constituency/Phrase Structure (PS) representation of sentence structure. However, recently, we observe a steady increase in popularity of Dependency Structure (DS) formalisms. Several researchers have compared the two alternatives, in terms of linguistic adequacy (Nivre, 2005; Schneider, 2008), practical applications (Ding and Palmer, 2005), and evaluations (Lin, 1995).

Dependency theory is historically accredited to Lucien Tesnière (1959), although the relation of dependency between words was only one of the various key elements proposed to represent sentence structures. In fact, the original formulation incorporates the notion of *chunk*, as well as a special type of structure to represent *coordination*.

The Tesnière Dependency Structure (TDS) representation we propose in (Sangati and Mazza, 2009), is an attempt to formalize the original work of Tesnière, with the intention to develop a simple but consistent representation which combines constituencies and dependencies. As part of this work, we have implemented an automatic conversion[1] of the English Penn Wall Street Journal (WSJ) treebank into the new annotation scheme.

In the current work, after introducing the key elements of TDS (section 2), we describe a first probabilistic extension to this framework, which aims at modeling the different levels of the representation (section 3). We test our model on parsing the WSJ treebank using a re-ranking framework. This technique allows us to efficiently test our system without needing a specialized parser, and to use the standard evaluation metric on the original PS version of the treebank. In section 3.4 we also introduce new evaluation schemes on specific aspects of the new TDS representation which we will include in the results presented in section 3.4.

2 TDS representation

It is beyond the scope of this paper to provide an exhaustive description of the TDS representation of the WSJ. It is nevertheless important to give the reader a brief summary of its key elements, and compare it with some of the other representations of the WSJ which have been proposed. Figure 1 shows the original PS of a WSJ tree (a), together with 3 other representations: (b) TDS, (c) DS[2], and (d) CCG (Hockenmaier and Steedman, 2007).

[1] staff.science.uva.nl/~fsangati/TDS

[2] The DS representation is taken from the conversion procedure used in the CoNLL 2007 Shared Task on dependency parsing (Nivre et al., 2007). Although more elaborate representation have been proposed (de Marneffe and Manning, 2008; Cinková et al., 2009) we have chosen this DS representation because it is one of the most commonly used within the CL community, given that it relies on a fully automatic conversion procedure.

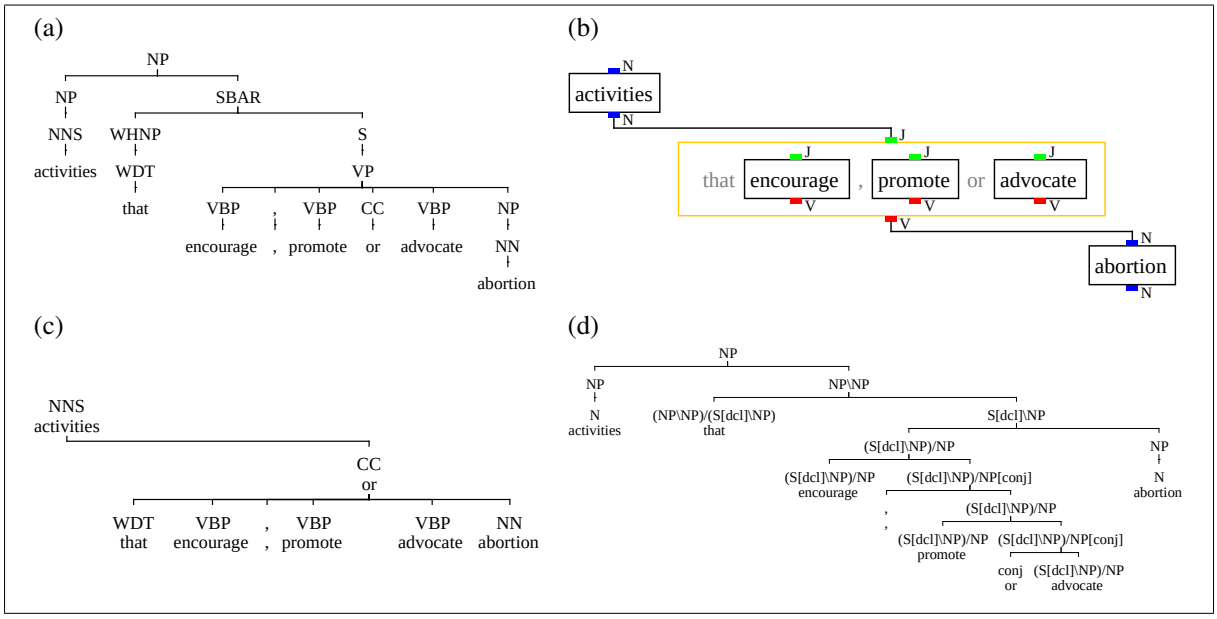

Figure 1: Four different structure representations, derived from a sentence of the WSJ treebank (section 00, #977). (a) PS (original), (b) CCG, (c) DS, (d) TDS.

Words and Blocks In TDS, words are divided in *functional* words (determiners, prepositions, etc.) and *content* words (verbs, nouns, etc.). *Blocks* are the basic elements (chunks) of a structure, which can be combined either via the dependency relation or the junction operation. Blocks can be of two types: *standard* and *junction* blocks. Both types may contain any sequence of functional words. Standard blocks (depicted as black boxes) represent the elementary chunks of the original PS, and include exactly one content word.

Coordination Junction blocks (depicted as yellow boxes) are used to represent coordinated structures. They contain two or more blocks (conjuncts) possibly coordinated by means of functional words (conjunctions). In Figure 1(d) the yellow junction block contains three separate standard blocks. This representation allows to capture the fact that these conjuncts occupy the same role: they all share the relativizer 'that', they all depend on the noun 'activities', and they all govern the noun 'abortion'. In Figure 1(a,c), we can notice that both PS and DS do not adequately represent coordination structures: the PS annotation is rather flat, avoiding to group the three verbs in a unique unit, while in the DS the last noun 'abortion' is at the same level of the verbs it should be a dependent of. On the other hand, the CCG structure of Figure 1(d), properly represents the coordination. It does so by grouping the first three verbs in a unique constituent which is in turn binarized in a right-branching structure. One of the strongest advantages of the CCG formalism, is that every structure can be automatically mapped to a logical-form representation. This is one reason why it needs to handle coordinations properly. Nevertheless, we conjecture that this representation of coordination might introduce some difficulties for parsing: it is very hard to capture the relation between 'advocate' and 'abortion' since they are several levels away in the structure.

Categories and Transference There are 4 different block categories, which are indicated with little colored bricks (as well as one-letter abbreviation) on top and at the bottom of the corresponding blocks: verbs (red, V), nouns (blue, N), adverbs (yellow, A), and adjectives (green, J). Every block displays at the bottom the *original category* determined by the content word (or the original category of the conjuncts if it is a junction structure), and at the top, the *derived category* which relates to the grammatical role of the whole block in relation to the governing block. In several cases we can observe a shift in the categories of a block, from the original to the derived category. This phenomenon is called *transference* and often occurs by means of functional words in the block. In Figure 1(b) we can observe the transference of the junction block, which has the original category of a verb, but takes the role of an adjective (through the relativizer 'that') in modifying the noun 'activities'.

$$P(S) = P_{BGM}(S) \cdot P_{BEM}(S) \cdot P_{WFM}(S) \quad (1)$$

$$P_{BGM}(S) = \prod_{B \in dependentBlocks(S)} P(B|parent(B), direction(B), leftSibling(B)) \quad (2)$$

$$P_{BEM}(S) = \prod_{B \in blocks(S)} P(elements(B)|derivedCat(B)) \quad (3)$$

$$P_{WFM}(S) = \prod_{B \in standardBlocks(S)} P(cw(B)|cw(parent(B)), cats(B), fw(B), context(B)) \quad (4)$$

Table 1: Equation (1) gives the likelihood of a structure S as the product of the likelihoods of generating three aspects of the structure, according to the three models (BGM, BEM, WFM) specified in equations (2-4) and explained in the main text.

3 A probabilistic Model for TDS

This section describes the probabilistic generative model which was implemented in order to disambiguate TDS structures. We have chosen the same strategy we have described in (Sangati et al., 2009). The idea consists of utilizing a state of the art parser to compute a list of k-best candidates of a test sentence, and evaluate the new model by using it as a reranker. How well does it select the most probable structure among the given candidates? Since no parser currently exists for the TDS representation, we utilize a state of the art parser for PS trees (Charniak, 1999), and transform each candidate to TDS. This strategy can be considered a first step to efficiently test and compare different models before implementing a full-fledged parser.

3.1 Model description

In order to compute the probability of a given TDS structure, we make use of three separate probabilistic generative models, each responsible for a specific aspect of the structure being generated. The probability of a TDS structure is obtained by multiplying its probabilities in the three models, as reported in the first equation of Table 1.

The first model (equation 2) is the **Block Generation Model (BGM)**. It describes the event of generating a block B as a dependent of its parent block (governor). The dependent block B is identified with its categories (both original and derived), and its functional words, while the parent block is characterized by the original category only. Moreover, in the conditioning context we specify the direction of the dependent with respect to the parent[3], and its adjacent left sister (*null* if not present) specified with the same level of details of B. The model applies only to dependent blocks[4].

The second model (equation 3) is the **Block Expansion Model (BEM)**. It computes the probability of a generic block B of known derived category, to expand to the list of elements it is composed of. The list includes the category of the content word, in case the expansion leads to a standard block. In case of a junction structure, it contains the conjunctions and the conjunct blocks (each identified with its categories and its functional words) in the order they appear. Moreover, all functional words in the block are added to the list[5]. The model applies to all blocks.

The third model (equation 4) is the **Word Filling Model (WFM)**, which applies to each standard block B of the structure. It models the event of filling B with a content word (cw), given the content word of the governing block, the categories ($cats$) and functional words (fw) of B, and further information about the context[6] in which B occurs. This model becomes particularly interest-

[3] A dependent block can have three different positions with respect to the parent block: left, right, inner. The first two are self-explanatory. The *inner* case occurs when the dependent block starts after the beginning of the parent block but ends before it (e.g. *a nice dog*).

[4] A block is a dependent block if it is not a conjunct. In other words, it must be connected with a line to its governor.

[5] The attentive reader might notice that the functional words are generated twice (in BGM and BEM). This decision, although not fully justified from a statistical viewpoint, seems to drive the model towards a better disambiguation.

[6] $context(B)$ comprises information about the grandparent block (original category), the adjacent left sibling block (derived category), the direction of the content word with respect to its governor (in this case only left and right), and the absolute distance between the two words.

ing when a standard block is a dependent of a junction block (such as 'abortion' in Figure 1(d)). In this case, the model needs to capture the dependency relation between the content word of the dependent block and each of the content words belonging to the junction block[7].

3.2 Smoothing

In all the three models we have adopted a smoothing techniques based on back-off level estimation as proposed by Collins (1999). The different back-off estimates, which are listed in decreasing levels of details, are interpolated with confidence weights[8] derived from the training corpus.

The first two models are implemented with two levels of back-off, in which the last is a constant value (10^{-6}) to make the overall probability small but not zero, for unknown events.

The third model is implemented with three levels of back-off: the last is set to the same constant value (10^{-6}), the first encodes the dependency event using both pos-tags and lexical information of the governor and the dependent word, while the second specifies only pos-tags.

3.3 Experiment Setup

We have tested our model on the WSJ section of Penn Treebank (Marcus et al., 1993), using sections 02-21 as training and section 22 for testing. We employ the Max-Ent parser, implemented by Charniak (1999), to generate a list of k-best PS candidates for the test sentences, which are then converted into TDS representation.

Instead of using Charniak's parser in its original settings, we train it on a version of the corpus in which we add a special suffix to constituents which have circumstantial role[9]. This decision is based on the observation that the TDS formalism well captures the argument structure of verbs, and we believe that this additional information might benefit our model.

We then applied our probabilistic model to re-rank the list of available k-best TDS, and evaluate the selected candidates using several metrics which will be introduced next.

3.4 Evaluation Metrics for TDS

The re-ranking framework described above, allows us to keep track of the original PS of each TDS candidate. This provides an implicit advantage for evaluating our system, viz. it allows us to evaluate the re-ranked structures both in terms of the standard evaluation benchmark on the original PS (F-score) as well as on more refined metrics derived from the converted TDS representation. In addition, the specific head assignment that the TDS conversion procedure performs on the original PS, allows us to convert every PS candidate to a standard projective DS, and from this representation we can in turn compute the standard benchmark evaluation for DS, i.e. unlabeled attachment score[10] (UAS) (Lin, 1995; Nivre et al., 2007).

Concerning the TDS representation, we have formulated 3 evaluation metrics which reflect the accuracy of the chosen structure with respect to the gold structure (the one derived from the manually annotated PS), regarding the different components of the representation:

Block Detection Score (BDS): the accuracy of detecting the correct boundaries of the blocks in the structure[11].

Block Attachment Score (BAS): the accuracy of detecting the correct governing block of each block in the structure[12].

Junction Detection Score (JDS): the accuracy of detecting the correct list of content-words composing each junction block in the structure[13].

[7] In order to derive the probability of this multi-event we compute the average between the probabilities of the single events which compose it.

[8] Each back-off level obtains a confidence weight which decreases with the increase of the *diversity of the context* ($\theta(C_i)$), which is the number of separate events occurring with the same context (C_i). More formally if $f(C_i)$ is the frequency of the conditioning context of the current event, the weight is obtained as $f(C_i)/(f(C_i) \cdot \mu \cdot \theta(C_i))$; see also (Bikel, 2004). In our model we have chosen μ to be 5 for the first model, and 50 for the second and the third.

[9] Those which have certain function tags (e.g. ADV, LOC, TMP). The full list is reported in (Sangati and Mazza, 2009). It was surprising to notice that the performance of this slightly modified parser (in terms of F-score) is only slightly lower than how it performs out-of-the-box (0.13%).

[10] UAS measures the percentage of words (excluding punctuation) having the correct governing word.

[11] It is calculated as the harmonic mean between recall and precision between the test and gold set of blocks, where each block is identified with two numerical values representing the start and the end position (punctuation words are discarded).

[12] It is computed as the percentage of words (both functional and content words, excluding punctuation) having the correct governing block. The governing block of a word, is defined as the governor of the block it belongs to. If the block is a conjunct, its governing block is computed recursively as the governing block of the junction block it belongs to.

[13] It is calculated as the harmonic mean between recall and precision between the test and gold set of junction blocks expansions, where each expansion is identified with the list of content words belonging to the junction block. A recursive junction structure expands to a list of lists of content-words.

	F-Score	UAS	BDS	BAS	JDS
Charniak ($k = 1$)	89.41	92.24	94.82	89.29	75.82
Oracle Best F-Score ($k = 1000$)	97.47	96.98	97.03	95.79	82.26
Oracle Worst F-Score ($k = 1000$)	57.04	77.04	84.71	70.10	43.01
Oracle Best JDS ($k = 1000$)	90.54	93.77	96.20	90.57	93.55
PCFG-reranker ($k = 5$)	89.03	92.12	**94.86**	88.94	**75.88**
PCFG-reranker ($k = 1000$)	83.52	87.04	92.07	82.32	69.17
TDS-reranker ($k = 5$)	**89.65**	**92.33**	94.77	**89.35**	**76.23**
TDS-reranker ($k = 10$)	89.10	92.11	94.58	88.94	75.47
TDS-reranker ($k = 100$)	86.64	90.24	93.11	86.34	69.60
TDS-reranker ($k = 500$)	84.94	88.62	91.97	84.43	65.30
TDS-reranker ($k = 1000$)	84.31	87.89	91.42	83.69	63.65

Table 2: Results of Charniak's parser, the TDS-reranker, and the PCFG-reranker according to several evaluation metrics, when the number k of best-candidates increases.

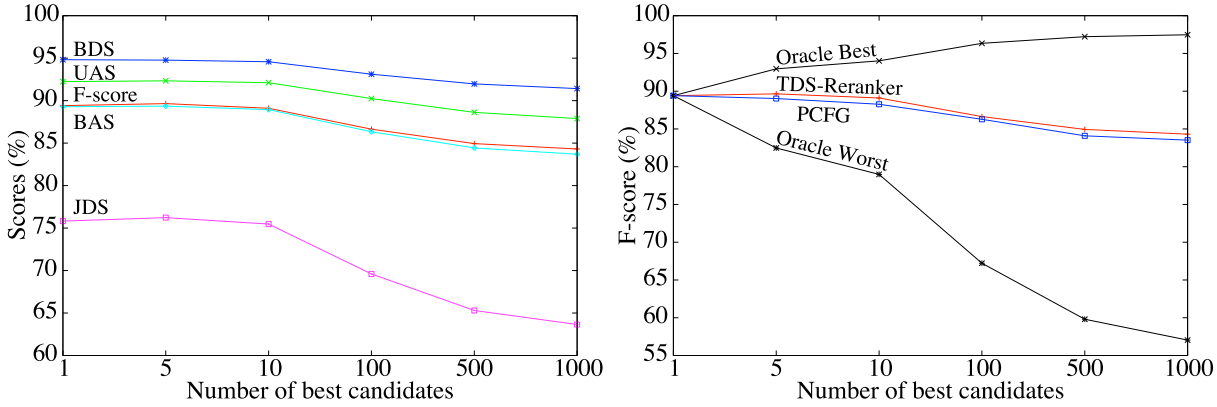

Figure 2: **Left**: results of the TDS-reranking model according to several evaluation metrics as in Table 2. **Right:** comparison between the F-scores of the TDS-reranker and a vanilla PCFG-reranker (together with the lower and the upper bound), with the increase of the number of best candidates.

3.5 Results

Table 2 reports the results we obtain when re-ranking with our model an increasing number of k-best candidates provided by Charniak's parser (the same results are shown in the left graph of Figure 2). We also report the results relative to a PCFG-reranker obtained by computing the probability of the k-best candidates using a standard vanilla-PCFG model derived from the same training corpus. Moreover, we evaluate, by means of an oracle, the upper and lower bound of the F-Score and JDS metric, by selecting the structures which maximizes/minimizes the results.

Our re-ranking model performs rather well for a limited number of candidate structures, and outperforms Charniak's model when $k = 5$. In this case we observe a small boost in performance for the detection of junction structures, as well as for all other evaluation metrics, except for the BDS.

The right graph in Figure 2 compares the F-score performance of the TDS-reranker against the PCFG-reranker. Our system consistently outperforms the PCFG model on this metric, as for UAS, and BAS. Concerning the other metrics, as the number of k-best candidates increases, the PCFG model outperforms the TDS-reranker both according to the BDS and the JDS.

Unfortunately, the performance of the re-ranking model worsens progressively with the increase of k. We find that this is primarily due to the lack of robustness of the model in detecting the block boundaries. This suggests that the system might benefit from a separate preprocessing step which could chunk the input sentence with higher accuracy (Sang et al., 2000). In addition the same module could detect local (intra-clausal) coordinations, as illustrated by (Marinčič et al., 2009).

4 Conclusions

In this paper, we have presented a probabilistic generative model for parsing TDS syntactic representation of English sentences. We have given evidence for the usefulness of this formalism: we consider it a valid alternative to commonly used PS and DS representations, since it incorporates the most relevant features of both notations; in addition, it makes use of junction structures to represent coordination, a linguistic phenomena highly abundant in natural language production, but often neglected when it comes to evaluating parsing resources. We have therefore proposed a special evaluation metrics for junction detection, with the hope that other researchers might benefit from it in the future. Remarkably, Charniak's parser performs extremely well in all the evaluation metrics besides the one related to coordination.

Our parsing results are encouraging: the overall system, although only when the candidates are highly reliable, can improve on Charniak's parser on all the evaluation metrics with the exception of chunking score (BDS). The weakness on performing chunking is the major factor responsible for the lack of robustness of our system. We are considering to use a dedicated pre-processing module to perform this step with higher accuracy.

Acknowledgments The author gratefully acknowledge funding by the Netherlands Organization for Scientific Research (NWO): this work is funded through a Vici-grant "Integrating Cognition" (277.70.006) to Rens Bod. We also thank 3 anonymous reviewers for very useful comments.

References

Daniel M. Bikel. 2004. Intricacies of Collins' Parsing Model. *Comput. Linguist.*, 30(4):479–511.

Eugene Charniak. 1999. A Maximum-Entropy-Inspired Parser. Technical report, Providence, RI, USA.

Silvie Cinková, Josef Toman, Jan Hajič, Kristýna Čermáková, Václav Klimeš, Lucie Mladová, Jana Šindlerová, Kristýna Tomšů, and Zdeněk Žabokrtský. 2009. Tectogrammatical Annotation of the Wall Street Journal. *The Prague Bulletin of Mathematical Linguistics*, (92).

Michael J. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford Typed Dependencies Representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, Manchester, UK.

Yuan Ding and Martha Palmer. 2005. Machine Translation Using Probabilistic Synchronous Dependency Insertion Grammars. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 541–548.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Dekang Lin. 1995. A Dependency-based Method for Evaluating Broad-Coverage Parsers. In *In Proceedings of IJCAI-95*, pages 1420–1425.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Domen Marinčič, Matjaž Gams, and Tomaž Šef. 2009. Intraclausal Coordination and Clause Detection as a Preprocessing Step to Dependency Parsing. In *TSD '09: Proceedings of the 12th International Conference on Text, Speech and Dialogue*, pages 147–153, Berlin, Heidelberg. Springer-Verlag.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan Mcdonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 Shared Task on Dependency Parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic.

Joakim Nivre. 2005. Dependency Grammar and Dependency Parsing. Technical report, Växjö University: School of Mathematics and Systems Engineering.

Erik F. Tjong Kim Sang, Sabine Buchholz, and Kim Sang. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal*.

Federico Sangati and Chiara Mazza. 2009. An English Dependency Treebank à la Tesnière. In *The 8th International Workshop on Treebanks and Linguistic Theories*, pages 173–184, Milan, Italy.

Federico Sangati, Willem Zuidema, and Rens Bod. 2009. A generative re-ranking model for dependency parsing. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 238–241, Paris, France, October.

Gerold Schneider. 2008. *Hybrid long-distance functional dependency parsing*. Ph.D. thesis, University of Zurich.

Lucien Tesnière. 1959. *Eléments de syntaxe structurale*. Editions Klincksieck, Paris.