Improved Parsing for Argument-Clusters Coordination

Jessica Ficler
Computer Science Department
Bar-Ilan University
Israel
jessica.ficler@gmail.com

Yoav Goldberg
Computer Science Department
Bar-Ilan University
Israel
yoav.goldbreg@gmail.com

Abstract

Syntactic parsers perform poorly in prediction of Argument-Cluster Coordination (ACC). We change the PTB representation of ACC to be more suitable for learning by a statistical PCFG parser, affecting 125 trees in the training set. Training on the modified trees yields a slight improvement in EVALB scores on sections 22 and 23. The main evaluation is on a corpus of 4th grade science exams, in which ACC structures are prevalent. On this corpus, we obtain an impressive $\times 2.7$ improvement in recovering ACC structures compared to a parser trained on the original PTB trees.

1 Introduction

Many natural language processing systems make use of syntactic representations of sentences. These representations are produced by parsers, which often produce incorrect analyses. Many of the mistakes are in coordination structures, and structures involving non-constituent coordination, such as Argument Cluster Coordination, Right Node-Raising and Gapping (Dowty, 1988), are especially hard.

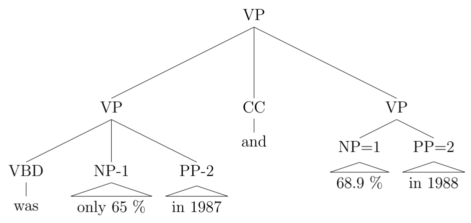
Coordination is a common syntactic phenomena and work has been done to improve coordination structures predication in the general case (Hogan, 2007; Hara et al., 2009; Shimbo and Hara, 2007; Okuma et al., 2009). In this work we focus on one particular coordination structure: Argument Cluster Coordination (ACC). While ACC are not common in the Penn TreeBank (Marcus et al., 1993), they commonly appear in other corpora. For example, in a dataset of questions from the Regents 4th grade science exam (the Aristo Challenge), 14% of the sentences include ACC.

ACC is characterized by non-constituent sequences that are parallel in structure. For instance, in *"I bought John a microphone on Monday and Richie a guitar on Saturday"*, the conjunction is between *"John a microphone on Monday"* and *"Richie a guitar on Saturday"* which are both non-constituents and include parallel arguments: the NPs *"John"* and *"Richie"*; the NPs *"a microphone"* and *"a guitar"*; and the PPs *"on Monday"* and *"on Saturday"*.

Previous NLP research on the Argument Clusters Coordination (Mouret, 2006) as well as the Penn TreeBank annotation guidelines (Marcus et al., 1993; Bies et al., 1995) focused mainly on providing representation schemes capable of expressing the linguistic nuances that may appear in such coordinations. The resulting representations are relatively complex, and are not easily learnable by current day parsers, including parsers that refine the grammar by learning latent annotations (Petrov et al., 2006), which are thought to be more agnostic to the annotations scheme of the trees. In this work, we suggest an alternative, simpler representation scheme which is capable of representing most of the Argument Cluster coordination cases in the Penn Treebank, and is better suited for training a parser. We show that by changing the annotation of 125 trees, we get a parser which is substantially better at handling ACC structures, and is also marginally better at parsing general sentences.

2 Arguments Cluster Coordination in the Penn Tree Bank

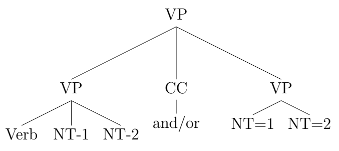
Argument Cluster Coordinations are represented in the PTB with two or more conjoined VPs, where the first VP contains a verb and indexed arguments, and the rest of the VPs lack a verb and include arguments with indices corresponding to those of the first conjoined VP. For example, consider the PTB representation of *"The Q ratio was only 65% in 1987 and 68.9% in 1988"*:



The main VP includes two conjoined VPs. The first VP includes the verb *was* and two indexed arguments: *"only 65%"* (1) and *"in 1987"* (2). The second VP does not include a verb, but only two arguments, that are co-indexed with the parallel argument at the first conjoined VP.

ACC structures in the PTB may include modifiers that are annotated under the main VP, and the conjoined VPs may includes arguments that are not part of the cluster. These are annotated with no index, i.e. *"insurance costs"* in [1a].

ACC structures are not common in the PTB. The training set includes only 141 ACC structures of which are conjoined by *and* or *or*. Some of them are complex but most (78%) have the following pattern (NT is used to denote non-terminals):



These structures can be characterized as follows: (1) the first token of the first conjoined VP is a verb; (2) the indexed arguments are direct children of the conjoined VPs; (3) the number of the indexed arguments is the same for each conjoined VP.

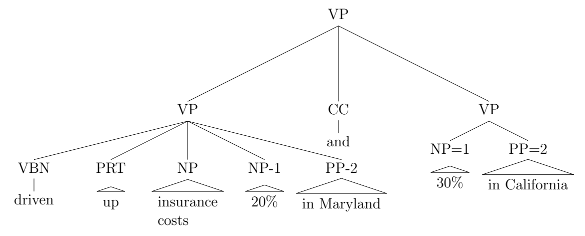
Almost all of these cases (98%) are symmetric: each of the conjoined VPs has the same types of indexed arguments. Non-symmetric clusters (e.g. "He made [these gestures]$^1_{NP}$ [to the red group]$^2_{PP}$ and [for us]$^2_{PP}$ [nothing]$^1_{NP}$") exist but are less common.

We argue that while the PTB representation for ACC gives a clear structure and covers all the ACC forms, it is not a good representation for learning PCFG parsers from. The arguments in the clusters are linked via co-indexation, breaking the context-free assumptions that PCFG parsers rely on. PCFG parsers ignore the indexes, essentially losing all the information about the ACC construction. Moreover, ignoring the indexes result in "weird" CFG rules such as `VP` → `NP PP`. Not only that the RHS of these rules do not include a verbal component, it is also a very common structure for NPs. This makes the parser very likely to either mis-analyze the argument cluster as a noun-phrase, or to analyze some NPs as (supposedly ACC) VPs. The parallel nature of the construction is also lost. To improve the parser performance for ACC structures prediction, we suggest an alternative constituency representation for ACC phrases which is easier to learn.

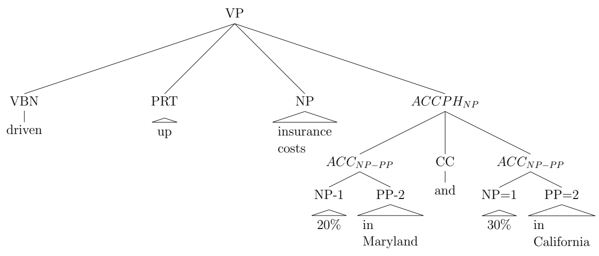
3 Alternative Representation for ACC

Our proposed representation for ACC respects the context-free nature of the parser. In order to avoid incorrect syntactic derivations and derivations that allows conjoining of clusters with other phrases, as well as to express the symmetry that occur in many ACC phrases, we change the PTB representation for ACC as follows: (1) we move the verb and non-indexed elements out of the first argument cluster to under the main VP; (2) each argument cluster is treated as a phrase, with new non-terminal symbols specific to argument clusters; (3) the conjunction of clusters also receives a dedicated phrase level. For example see comparison between the original and new representations:

[1]



(a) PTB representation

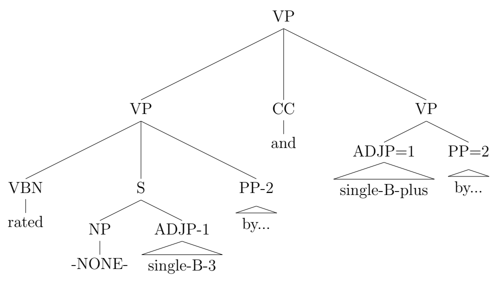


(b) Our modified tree

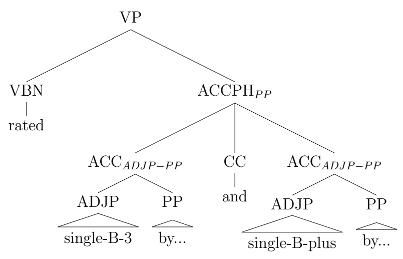
The main verb *driven* as well as the particle *up* and the non-indexed argument *insurance costs* are moved to the external VP. The two argument clusters (formerly VPs) receive dedicated phrase labels ACC_X, where X reflects the syntactic types of the indexed elements (e.g. ACC_{NP-PP} for the first cluster in [1b] above). The most common cases are ACC_{NP-PP} which appears in 41.6% of the clusters, $ACC_{ADJP-PP}$ with 21.2% of the clusters and ACC_{PP-PP} with 5.3% of the clusters.

Finally, we introduce a new phrase type ($ACCPH_X$) for the coordination of the two clusters. Here X denotes the main element in the clusters, determined heuristically by taking the first of the following types that appear in any of the clusters: NP, PP, ADJP, SBAR. Cases where the clusters contains an ADVP element are usually special (e.g. the following structure is missing "people" in the second cluster: ((NP 8000 people) (in Spain)) and ((NP 2000) (ADVP abroad))). For such cases, we add "ADVP" to the $ACCPH$ level label. Table 1 lists the $ACCPH$ level labels and their number of the appearances in the 125 modified trees.[1]

The representation is capable of representing common cases of ACC where the cluster elements are siblings. We similarly handle also some of the more complex cases, in which an extra layer appears between an indexed argument and the conjoined VP to host an empty element, such as in the following case with an extra S layer above *single-B-3*:



in which we remove the empty NP as well as the extra S layer:



[1]Parsers that apply latent annotations to the grammar, such as the Berkeley Parser (Petrov et al., 2006) we use in our experiments, can potentially learn some of our proposed refinements on their own. However, as we show in the experiments section, the performance of the Berkeley Parser on ACC structures significantly improve when applying our transformations prior to training.

Label	**#**	**Label**	**#**
$ACCPH_{NP}$	69	$ACCPH_{NP-ADVP}$	6
$ACCPH_{PP}$	36	$ACCPH_{PP-ADVP}$	11
$ACCPH_{ADJP}$	2	$ACCPH_{SBAR-ADVP}$	1

Table 1: The labels for the new level in the ACC trees. #: number of occurrences.

Limitations Our representation is similar to the representation that was suggested for ACC by Huddleston et al. (2002) in their comprehensive linguistic description of the English grammar. However, while it is capable of representing the common cases of ACC, it does not cover some complex and rare cases encountered in the PTB: (1) Argument-Cluster structures that include errors such as missing indexed argument and a wrong POS tag for the main verb; (2) ACC constructions where the main verb is between the indexed arguments such as the following: "([About half]$_1$ invested [in government bonds]$_2$) and ([about 10%]$_1$ [in cash]$_2$)"; (3) Argument-Cluster structures that include an indexed phrase which is not a direct child of the cluster head and has non-empty siblings, such as in the following case that includes an indexed argument (*8%*) which is not directly under the conjoined VP and has non-empty sibling (*of*): *"see a raise [[of] [8%]$_{NP-1}$]$_{PP}$ in the first year] and [7%]$_{NP=1}$ in each of the following two years"*.

Our changes are local and appear in small number of trees (0.003% of the PTB train set). We also ignore more complex cases of ACC. Yet, training the parser with the modified trees significantly improves the parser results on ACC structures.

4 Experiments

We converted 125 trees with ACC structures in the training sets (sections 2-21) of the PTB to the new representation, and trained the Berkeley parser (Petrov et al., 2006) with its default settings.

As the PTB test and dev sets have only 12 ACC structures that are coordinated by *and* or *or*, we evaluate the parser on Regents, a dataset in which ACC structures are prevalent (details below). As Regents does not include syntactic structures, we focus on the ACC phenomena and evaluate the parsers' ability to correctly identify the spans of the clusters and the arguments in them.

To verify that the new representation does not harm general parsing performance, we also eval-

Dataset		**R**	**P**	**F1**
Dev	PTB Trees	90.88	90.89	90.88
	Modified Trees	90.97	91.21	91.09
Test	PTB Trees	90.36	90.79	90.57
	Modified Trees	90.62	91.06	90.84

Table 2: Parsing results (EVALB) on PTB Sections 22 (DEV) and 23 (TEST).

	PTB Trees	**Modified Trees**
ACC_{PTB}	13.0	-
ACC_{OUR}	24.1	64.8

Table 3: The parser Recall score in recovering ACC conjunct spans on the Regents dataset. ACC_{PTB}: the set is annotated with the verb inside the first cluster. ACC_{OUR}: the set is annotated following our approach.

uate the parer on the traditional development and test sets (sections 22 and 23). As can be seen in Table 2, the parser results are slightly better when trained with the modified trees.[2]

4.1 Regents data-set

Regents – a dataset of questions from the Regents 4th grade science exam (the Aristo Challenge),[3] includes 281 sentences with coordination phrases, where 54 of them include Argument Cluster coordination. We manually annotated the sentences by marking the conjuncts spans for the constituent coordination phrases, e.g.:

Wendy (ran 19 miles) and (walked 9 miles)

as well as the spans of each component of the argument-cluster coordinations, including the inner span of each argument:

Mary paid ([$11.08] [for berries]) , ([$14.33] [for apples]) , and ([$9.31] [for peaches])

The bracketing in this set follow our proposed ACC bracketing, and we refer to it as ACC_{OUR}.

We also created a version in which the bracketing follow the PTB scheme, with the verb included in span of the first cluster, e.g.:

Mary ([paid] [$11.08] [for berries]) , ([$14.33] [for apples]) , and ([$9.31] [for peaches])

We refer to this dataset as ACC_{PTB}.

[2]The same trend holds also if we exclude the 12 modified trees from the evaluation sets.

[3]http://allenai.org/content/data/Regents.zip

We evaluate the parsers' ability to correctly recover the components of the coordination structures by computing the percentage of gold annotated phrases where the number of predicted conjunct is correct and all conjuncts spans (round brackets) are predicted correctly (Recall). For example, consider the following gold annotated phrase:

A restaurant served (9 pizzas during lunch) and (6 during dinner) today

A prediction of (*"9 pizzas during lunch"*, *"6 during dinner today"*) is considered as incorrect because the second conjunct boundaries are not matched to the gold annotation.

We compare the Recall score that the parser achieves when it is trained on the modified trees to the score when the parser is trained on the PTB trees.

When evaluated on all coordination cases in the Regents dataset (both ACC and other cases of constituent coordination), the parser trained on the modified trees was successful in recovering 54.3% of the spans, compared to only 47% when trained on the original PTB trees.

We now focus on specifically on the ACC cases (Table 3). When evaluating the PTB-trained parser on ACC_{PTB}, it correctly recovers only 13% of the ACC boundaries. Somewhat surprisingly, the PTB-trained parser performs *better* when evaluated against ACC_{OUR}, correctly recovering 24.1% of the structures. This highlights how unnatural the original ACC representation is for the parser: it predicts the alternative representation more often than it predicts the one it was trained on. When the parser is trained on the modified trees, results on ACC_{OUR} jump to 64.8%, correctly recovering $\times$2.7 more structures.

The previous results were on recovering the spans of the coordinated elements (the round brackets in the examples above). When measuring the Recall in recovering any of the arguments themselves (the elements surrounded by square brackets), the parser trained on the modified trees recovers 72.46% of the arguments in clusters, compared to only 58.29% recovery by the PTB-trained parser. We also measure in what percentage of the cases in which both the cluster boundaries (round brackets) were recovered correctly, all the internal structure (square brackets) was recovered correctly as well. The score is 80% when the parser trained on the modified trees compared to 61.5% when it is trained on the PTB-trees.

Overall, the parser trained on the modified trees significantly outperforms the one trained on the original trees in all the evaluation scenarios.

Another interesting evaluation is the ability of the parser that is trained on the modified trees to determine whether a coordination is of Argument Clusters type (that is, whether the predicted coordination spans are marked with the ACCPH label).[4] The results are a Recall of 57.4% and Precision of 83.78%. When we further require that both the head be marked as ACCPH and the internal structure be correct, the results are 48.14% Recall and 70.27% Precision.

5 Conclusions

By focusing on the details of a single and relatively rare syntactic construction, argument clusters coordination, we have been able to significantly improve parsing results for this construction, while also slightly improving general parsing results. More broadly, while most current research efforts in natural language processing and in syntactic parsing in particular is devoted to the design of general-purpose, data-agnostic techniques, such methods work on the common phenomena while often neglecting the very long tail of important constructions. This work shows that there are gains to be had also from focusing on the details of particular linguistic phenomena, and changing the data such that it is easier for a "data agnostic" system to learn.

Acknowledgments

This work was supported by The Allen Institute for Artificial Intelligence as well as the German Research Foundation via the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).

References

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing guidelines for treebank ii style penn treebank project. *University of Pennsylvania*, 97:100.

David Dowty. 1988. Type raising, functional composition, and non-constituent conjunction. In *Categorial grammars and natural language structures*, pages 153–197. Springer.

Kazuo Hara, Masashi Shimbo, Hideharu Okuma, and Yuji Matsumoto. 2009. Coordinate structure analysis with global structural constraints and alignment-based local features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 967–975. Association for Computational Linguistics.

Deirdre Hogan. 2007. Coordinate noun phrase disambiguation in a generative parsing model. Association for Computational Linguistics.

Rodney Huddleston, Geoffrey K Pullum, et al. 2002. The cambridge grammar of english. *Language. Cambridge: Cambridge University Press*, pages 1273–1362.

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.

François Mouret. 2006. A phrase structure approach to argument cluster coordination. In *Proceedings of the HPSG06 Conference*, pages 247–267. CSLI online Publications.

Hideharu Okuma, Kazuo Hara, Masashi Shimbo, and Yuji Matsumoto. 2009. Bypassed alignment graph for learning coordination in japanese sentences. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 5–8. Association for Computational Linguistics.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Masashi Shimbo and Kazuo Hara. 2007. A discriminative learning model for coordinate conjunctions. In *EMNLP-CoNLL*, pages 610–619.

[4]This measurement is relevant only when parsing based on our proposed annotation, and cannot be measured for parse trees based the original PTB annotation.