A Machine Learning Approach to Convert CCGbank to Penn Treebank[*]

Xiaotian Zhang[1,2] Hai Zhao[1,2†] Cong Hui[1,2]
(1) MOE-Microsoft Key Laboratory of Intelligent Computing and Intelligent System;
(2) Department of Computer Science and Engineering,
#800 Dongchuan Road, Shanghai, China, 200240
xtian.zh@gmail.com zhaohai@cs.sjtu.edu.cn huicong88126@gmail.com

Abstract
Conversion between different grammar frameworks is of great importance to comparative performance analysis of the parsers developed based on them and to discover the essential nature of languages. This paper presents an approach that converts Combinatory Categorial Grammar (CCG) derivations to Penn Treebank (PTB) trees using a maximum entropy model. Compared with previous work, the presented technique makes the conversion practical by eliminating the need to develop mapping rules manually and achieves state-of-the-art results.

Keywords: CCG, Grammar conversion.

[*] This work was partially supported by the National Natural Science Foundation of China (Grant No. 60903119 and Grant No. 61170114), the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20110073120022, the National Basic Research Program of China (Grant No. 2009CB320901), and the European Union Seventh Framework Program (Grant No. 247619).

[†] Corresponding author

1 Introduction

Much has been done in cross-framework performance analysis of parsers, and nearly all such analysis applies a converter across different grammar frameworks. Matsuzaki and Tsujii (2008) compared a Head-driven Phrase Structure Grammar (HPSG) parser with several Context Free Grammar (CFG) parsers by converting the parsing result to a shallow CFG analysis using an automatic tree converter based on Stochastic Synchronous Tree-Substitution Grammar. Clark and Curran (2007) converted the CCG dependencies of the CCG parser into those in Depbank by developing mapping rules via inspection so as to compare the performance of their CCG parser with the RASP parser. Performing such a conversion has been proven to be a time-consuming and non-trivial task (Clark and Curran, 2007).

Although CCGBank (Hockenmaier and Steedman, 2007) is a translation of the Penn Treebank (Marcus et al., 1993) into a corpus of Combinatory Categorial Grammar derivations, little work has been done in conversion from CCG derivations back to PTB trees besides (Clark and Curran, 2009) and (Kummerfeld et al., 2012). In the work of Clark and Curran (2009), they associated conversion rules with each local tree and developed 32 unary and 776 binary rule instances by manual inspection. Considerable time and effort were spent on the creation of these schemas. They show that although CCGBank is derived from PTB, the conversion from CCG back to PTB trees is far from trivial due to the non-isomorphic property of tree structures and the non-correspondence of tree labels. In the work of Kummerfeld et al. (2012), although no ad-hoc rules over non-local features were required, a set of instructions, operations, and also some special cases were defined.

Nevertheless, is it possible to convert CCG derivations to PTB trees using a statistical machine learning method instead of creating conversion grammars or defining instructions, and thus avoid the difficulties in developing these rules? Is it possible to restore the tree structure by only considering the change applied to the structure in the original generating process? Our proposed approach answers yes.

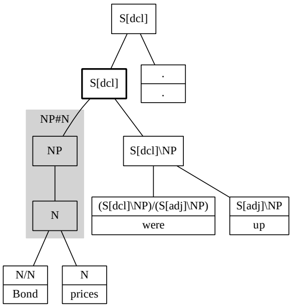
2 PTB2CCG and the Inverse

Let us first look at how CCG is derived from PTB. The basic procedure for converting a PTB tree to CCG described in (Hockenmaier and Steedman, 2007) consists of four steps:

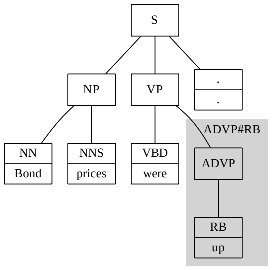
1. Determine constituent types;
2. Make the tree binary;
3. Assign categories;
4. Assign dependencies;

Clearly, only step 2 changes the tree structure by adding dummy nodes to make it "binary", precisely, to make every node have only one or two child nodes.

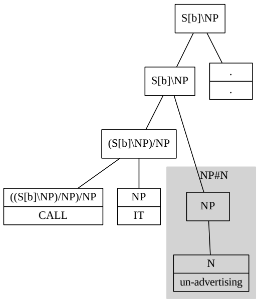
Take the short sentence "Bond prices were up" in Figure 1(a) and 1(b) for example. As a preprocessing step, combine every internal node which has only one child with its single child and label the new node with the catenation of the original labels and #. That means to combine the shaded nodes in Figure 1 and obtain the corresponding new node. After the preprocessing, the node with a bold border in Figure 1(a) can be identified as "dummy" because the structure of CCG turns out to be the same as that of PTB if that node is deleted and its children are attached directly to its parent.



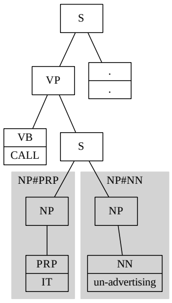
(a) Category A: CCG Example.



(b) Category A: PTB Example

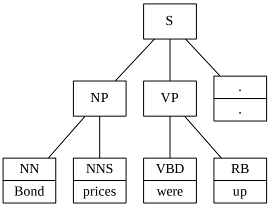


(c) Category B: CCG Example

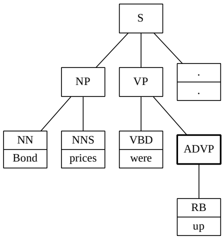


(d) Category B: PTB Example

Figure 1: CCG and corresponding PTB Examples of Category A and B. The grey shadow indicates the preprocessing that combines every internal node which has only one child with its single child. And the node with a bold border in Figure (a) is "dummy".



(a) Output PTB by $Classifier_1$



(b) Output PTB by $Classifier_2$

Figure 2: Output PTB by $Classifier_1$ and $Classifier_2$ when testing on the CCG sample in Figure 1(a). $Classifier_2$ corrects the output PTB of $Classifier_1$ by adding "ADVP", the node with a bold border, to dominate the leaf "RB".

However, it is more complicated in reality. There indeed exist CCG tree structures that can not be derived by just adding dummy nodes to the corresponding PTB trees, as shown in Figure 1(c) and 1(d), because actually a more sophisticated algorithm is used in the conversion from PTB to CCG. After investigation, all the cases can be grouped into two categories:

- Category A: The CCG tree structure can be derived from the PTB tree by adding dummy nodes, after the preprocessing step, as shown in Figure 1(a) and 1(b).
- Category B[1]: The CCG tree structure cannot be derived from the PTB tree by adding dummy nodes even if we perform the preprocessing, as shown in Figure 1(c) and 1(d).

The cases of Category B are less than 10% of the whole bank, so we simplify the problem by only focusing on solving the cases of Category A.

From the above discussion, after the preprocessing, one classifier is required to classify the CCG nodes into the target classes which are PTB labels and "dummy". However, preliminary experiments show that such a model still cannot handle the occasions in which a parent node occurs with a single child leaf in the PTB tree very well. For example, the S[adj]\NP node in the gold CCG tree (Figure 1(a)) tends to be classified as an RB node in the predicted PTB tree (Figure 2(a)) instead of the gold PTB label ADVP#RB (Figure 1(b)). So we use another classifier to predict whether each leaf node in the PTB tree produced by the first classifier has a parent dominating only itself and what the label of the parent is. This added classifier helps to identify the ADVP node (Figure 2(b)) in the above example and experiments show it increases the overall F-measure by up to 2%.

3 The Approach

Firstly, the training and test data need to be preprocessed as mentioned in Section 2. Then, classifiers are constructed based on the maximum entropy model[2]. As for the training process, the "dummy" nodes and the corresponding PTB labels could be identified by comparing post-order traversal sequences of each pair of CCG and PTB trees. And thus $Classifier_1$ can be trained. And based on the output of $Classifier_1$ tested on the training set and corresponding gold PTB trees, $Classifier_2$ can be trained.

When testing, there are two steps. Firstly, $Classifier_1$ is to classify the CCG labels into PTB labels and "dummy". The CCG tree is traversed in a bottom-up order and if the node is classified as "dummy", delete it and attach its children to its parent, otherwise replace the CCG label with the predicted PTB label. Then, an intermediate PTB tree is built. Secondly, $Classifier_2$ examines each leaf in the intermediate PTB tree and predicts whether a parent node should be added to dominate the leaf. The feature sets for the two classifiers are listed in Table 1.

4 Experiments

We evaluate the proposed approach on CCGBank and PTB. As in (Clark and Curran, 2009), we use Section 01-22 as training set[3], Section 00 as development set to tune the parameters and Section 23 as test set. Experiments are done separately with gold POS tags and auto POS tags predicted by Lapos tagger (Tsuruoka et al., 2011).

[1] According to our inspection, these cases always occur in certain language structures, such as complex objects, "of" phrases and so on.

[2] We use the implementation by Apache OpenNLP http://incubator.apache.org/opennlp/index.html

[3] Since the converter is trained on the gold banks, it doesn't have access to parsing output. And thus, the converting process is general and not specific to parser errors.

Feature Set 1	**Feature Set 2**
$Features_{CCGNode}$	WordForm
Label	POS
ParentLabel	ParentLabel
LSiblingLabel	IndexAmongChildren
LSiblingWordForm	ParentChildrenCnt
RSiblingLabel	LSiblingLabel
RSiblingWordForm	LSiblingWordForm
Children	RSiblingLabel
ChildCnt	RSiblingWordForm
$Features_{PTBNode}$	LSiblingRightMostLabel
Children	RSiblingLeftMostLabel
LSiblingLabel	SecondRSiblingLabel
POS	SecondLSiblingLabel
POS_{-1}	

Table 1: Features Used by Two Classifiers ("LSibling", and "RSibling" represent "Left sibling" and "Right sibling". POS_{-1} represents the POS tag of the previous word).

POS Tag	P	R	F
Gold	96.99	95.29	96.14
Lapos	96.82	95.12	95.96

Table 2: Evaluation on all the sentences of Category A in Section 23.

Firstly we leave out Category B and test on all the sentences belonging to Category A from Section 23. Table 2 shows our conversion accuracy. The numbers are bracketing precision, recall, F-score using the EVALB[4] evaluation script.

In order to compare with (Clark and Curran, 2009)[5], the approach is also tested on all of Section 00 and 23. The oracle conversion results are presented in Table 3. It shows the F-measure of our method is about one point higher than that of Clark and Curran (2009), no matter what kind of POS tags[6] is applied.

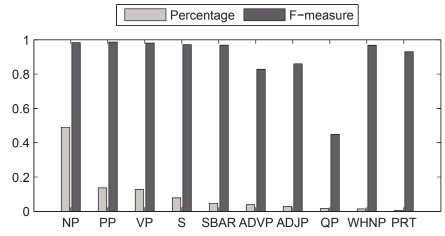


Figure 3: The conversion F-measure of top ten major kinds of phrases.

[4]http://nlp.cs.nyu.edu/evalb/

[5]We don't know which kind of POS tags, gold or predicted, was used in (Clark and Curran, 2009) as they did not report it in their paper.

[6]We have tried with different POS taggers and the results stay more or less stable.

	Section	P	R	F
Our$_{gold}$	00$_{all}$	**96.92**	94.82	**95.86**
	00$_{len \leq 40}$	**97.09**	95.40	**96.24**
	23$_{all}$	**96.67**	94.77	**95.71**
	23$_{len \leq 40}$	**96.69**	94.79	**95.73**
Our$_{lapos}$	00$_{all}$	96.85	94.74	95.79
	00$_{len \leq 40}$	97.03	95.34	96.18
	23$_{all}$	96.49	94.64	95.56
	23$_{len \leq 40}$	96.51	94.67	95.58
Clark& Curran, 2009	00$_{all}$	93.37	**95.15**	94.25
	00$_{len \leq 40}$	94.11	**95.65**	94.88
	23$_{all}$	93.68	**95.13**	94.40
	23$_{len \leq 40}$	93.75	**95.23**	94.48

Table 3: Evaluation on all of Section 00 and 23.

	-Simple	-Children	-Parent	-Sibling
F	93.8	91.28	95.18	**89.02**

Table 4: Results on Section 23 using gold POS tags and features excluding one category each time.

To investigate the effectiveness of the features, we divide them into four categories[7], children, parent, sibling features and simple features (label, POS and wordform), and test using feature sets from which one category of features is removed each time. Table 4 shows that all the kinds of features have an effect and the sibling-related features are the most effective.

In error analysis, the conversion F-measure of each kind of phrases is examined (see Figure 3). As for the F-measure of the top ten major kinds, seven of them are over 90%, two around 80% while the worst is 44.8%. The high accuracy in predicting most major kinds of phrases leads to the good overall performance of our approach and there is still some room to improve by further finding effective features to classify QP phrases better.

5 Conclusions

We have proposed a practical machine learning approach[8] to convert the CCG derivations to PTB trees efficiently and achieved an F-measure over 95%. The core of the approach is based on the maximum entropy model. Compared with conversion methods that use mapping rules, applying such a statistical machine learning method helps saving considerable time and effort.

References

Clark, S. and Curran, J. (2007). Formalism-independent parser evaluation with ccg and depbank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 248–255, Prague, Czech Republic. Association for Computational Linguistics.

Clark, S. and Curran, J. R. (2009). Comparing the accuracy of ccg and penn treebank parsers. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 53–56, Suntec, Singapore. Association for Computational Linguistics.

[7]Due to space limitations, we show the importance of every category of features instead of each individual feature.

[8] This software has been released, https://sourceforge.net/p/ccg2ptb/home/Home/.

Hockenmaier, J. and Steedman, M. (2007). Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33:355–396.

Kummerfeld, J. K., Klein, D., and Curran, J. R. (2012). Robust conversion of ccg derivations to phrase structure trees. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 105–109, Jeju Island, Korea. Association for Computational Linguistics.

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

Matsuzaki, T. and Tsujii, J. (2008). Comparative parser performance analysis across grammar frameworks through automatic tree conversion using synchronous grammars. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 545–552, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tsuruoka, Y., Miyao, Y., and Kazama, J. (2011). Learning with lookahead: can history-based models rival globally optimized models? In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, pages 238–246, Stroudsburg, PA, USA. Association for Computational Linguistics.