Generating IRTG grammars from parallel data

Evelin Ács Gábor Recski
Department of Automation and Applied Informatics
Budapest University of Technology and Economics
evelin@aut.bme.hu, recski@aut.bme.hu

Abstract. Graph transformations are utilized by several common tasks in natural language processing (NLP), in particular those that handle syntactic trees, dependency structures such as Universal Dependencies (UD), or semantic graphs such as AMR and 4lang. Interpreted Regular Tree Grammars (IRTGs) encode the correspondence between sets of such structures and have in recent years been used to perform both syntactic and semantic parsing. Such a grammar allows converting from any of the above algebraic structures to any or all of the others, e.g. generating English text from dependency graphs. It can also be trained on correspondences between grammatical and semantic structures and surface realizations. In this paper we present a method to generate an IRTG grammar from parallel data of syntactic trees and UD graphs, and also some preliminary experiments for generating 4lang graphs from the definitions of the Longman dictionary.

Keywords: Semantic parsing; Natural language processing; IRTG; Graph transformation

1 Introduction

For many complex NLP-related tasks, such as question answering, machine translation and dialogue systems, one of the most limiting factors is the absence of deep semantic parsing. State-of-the-art systems use supervised machine learning algorithms and encode the meaning of words in multidimensional vector spaces. However, the understanding of the structures of these representations is very limited. Another approach is to represent the meaning of natural language text using concept networks. As several other core NLP tasks, such as dependency parsing (which maps raw text to directed acyclic graphs over words of each input sentence) are instances of graph transformation, the whole task of end-to-end semantic parsing can be viewed as such and its functionality can be implemented using a single graph grammar.
This paper implements a program which generates an IRTG [1] using parallel data, which maps rules of a regular tree grammar to operations over raw text, phrase structure trees, UD [2] and 4lang [3] graphs, thereby allowing for efficient transformation between any of the representations. Our contribution is

available on GitHub under an MIT license.[1]
The paper is structured as follows: Section 2 gives a brief description of the syntactic and semantic representations used by the program and introduces IRTGs. Section 3 describes our IRTG generating program and in Section 4 we present some preliminary results.

2 Background

2.1 Syntactic representations

Constituency structure is equivalent to a bracketing of words of a sentence and can be represented as a tree structure, nodes of which are called constituents of the sentence, as in Figure 1.

The Penn Treebank ([4]) is a part-of-speech annotated dataset of American English sentences, consisting of over 4.5 million words.

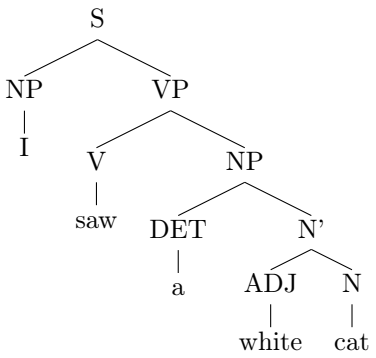

Figure 1: The structure of the sentence 'I saw a white cat.'

Another approach to represent syntactic structure are dependency graphs. The notion of dependency means that words (head and dependent) are connected to each other via directed links. The finite verb counts as the root, or the structural center of the sentence, while the relations between heads and dependents determine the structure, as in the example of Figure 2.

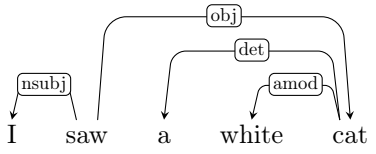

Figure 2: Dependency analysis of the sentence 'I saw a white cat.'

[1] https://github.com/evelinacs/semantic_parsing_with_IRTGs/tree/master/code/generate_grammar/template_based_grammar_generator

```
// 34312
NP -> _NP2_det_DT_NN(DT, NN) [0.08811233346686799]
[string] *(?1,?2)
[tree] NP2(?1, ?2)
[ud] merge(f_dep(merge("(r<root> :det (d<dep>))", r_dep(?1))),?2)
[fourlang] ?2
```

Figure 3: An example IRTG rule generated by our program

The Universal Dependencies (UD) project[2] ([2]) is a cross-linguistically consistent annotation system and treebanks for over 60 languages.

2.2 4lang

4lang is a formalism which builds directed graphs for semantic representation. In such graphs, nodes stand for concepts, which do not have any grammatical attributes, and contains shared knowledge of competent speakers. These can be connected via three types of edges, namely 0, 1 and 2 [3]. 0-edge represents attribution ($cat \xrightarrow{0} white$), the IS_A relation ($cat \xrightarrow{0} mammal$) and unary predication ($cat \xrightarrow{0} meow$). 1 and 2-edges connect binary predicates to their arguments ($John \xleftarrow{1} buy \xrightarrow{2} book$).

2.3 IRTGs

Interpreted regular tree grammars [5] are context-free grammars in which rules are simultaneously mapped to operations of an arbitrary number of algebras. Thus, an IRTG parser can accept inputs in any of the defined interpretations and can convert the input data into any of the other interpretations as output. In our contribution we use the `s-graph algebra` [6], which is for combining subgraphs and the `tag tree algebra` [7], which is a simple tree manipulation language. For a more detailed overview the reader is referred to [8].

3 Generating IRTG rules from parallel data

The code generates IRTG rules - based on several pre-defined parameterized templates - from NPs of the Penn Treebank and those parsed by the Stanford parser [9]. It compares the phrase structure tree and the dependency graph of each noun phrase. The input for the phrase structure tree data is in Penn Treebank format and the dependency graph data is extracted from the output of the Stanford parser (which is generated by pattern matching). The output of rule generation contains only unique rules in descending order of their occurrences. Figure 3 shows an example rule.

[2] `http://universaldependencies.org/`

```
ADJP -> _ADJP_unary_JJ(JJ)
[string] ?1
[tree] ADJP(?1)
[ud] ?1
[fourlang] ?1
```

Figure 4: A unary rule

The first line is a comment, containing the number of occurrences (i.e. the program found the relation described by the RTG line this many times in the input data). The second line is the RTG rule, and the remaining lines describe how this relation is realized in the corresponding algebras.

To generate the IRTG rules, the program iterates over the phrase structure trees and the corresponding dependency graphs from the input. When processing a tree-graph pair, first it finds the nodes which only have one child that is not a leaf node and generates unary rules from them. An example is shown in Figure 4.

Then it iterates over all dependencies. First it looks up the corresponding nodes of the head and dependent from the tree and finds their lowest common ancestor alongside with some additional information (how many children the ancestor has, what are the ancestor's immediate children leading to the current nodes, and in the case of a ternary ancestor, are these children next to each other).

If the common ancestor has three children, it determines which two adjacent children are connected by a dependency relation. This information is used to generate proper `merge` rules in the tag tree interpretation.

Next it checks whether the order of the nodes in the dependency graph is the same as in the constituency tree. Given the rigidity and ordered nature of the tag tree algebra, the rule generation is based on the phrase structure of a subtree. Based on this, the RTG line (its left-hand side, and the arguments on the right-hand side) is derived from the phrase structure tree, and the [tree] interpretation either simply reflects the node type of the head and the number of its children or a merge operation is performed between two subtrees. To generate the [ud] and [fourlang] interpretations, the order of the nodes in the phrase structure tree must be considered, as the direction of some edges in these graphs are reversed with regards to the order of nodes in the phrase structure tree, and this must be reflected in the generated rules. The edge types in the [fourlang] interpretation are derived from the UD edge types using a predefined mapping implemented before, as seen in Table 1.

Dependency	**Edge**
advcl	$w_1 \xrightarrow{0} w_2$
advmod	
amod	
nmod	
nummod	
appos	$w_1 \underset{0}{\overset{0}{\rightleftharpoons}} w_2$
dislocated	
csubj	$w_1 \underset{0}{\overset{1}{\rightleftharpoons}} w_2$
nsubj	
ccomp	$w_1 \xrightarrow{2} w_2$
obj	
xcomp	
The treatment of `case`	
case + nmod	$w_1 \xleftarrow{1} w_3 \xrightarrow{2} w_2$
case + nsubj	
case + obl	
English subtypes	
obl:npmod	$w_1 \xrightarrow{0} w_2$
nmod:tmod	$w_1 \xleftarrow{1} \texttt{AT} \xrightarrow{2} w_2$
obl:tmod	
nmod:poss	$w_2 \xleftarrow{1} \texttt{HAS} \xrightarrow{2} w_1$

Table 1: UD-conform version of the `dep_to_4lang` mapping ([8], upgraded from [10]).

4 Some preliminary results

4.1 Validation on the input data

Our grammar have been evaluated on trees with an NP as a root node and any node within a tree must have at most three children. This subset covers 84,8% of all NPs of the Wall Street Journal section of the Penn Treebank. The WSJ section contains 243 914 NPs of which 206 841 meet the aforementioned restrictions. Parsing this subset using the generated grammar resulted in 202 549 successful parses, which constitutes 97,9% of the test data and 83% of all NPs. We can transform structures in this subset from any interpretation to any other interpretation, including converting UD graphs to strings. These results were presented at MSZNY2019 ([8]).

4.2 4lang definitions

To test our grammar on noun phrases which are not in the input used to generate the grammar, we tried to extract meaning representations from the definitions of the Longman dictionary. It contains 30126 definitions. These have been parsed with the Stanford parser and the ones that were classified as NPs (8520), have been used to evaluate the grammar. 8465 definitions have been successfully parsed, but due to the non-deterministic nature of IRTGs, the first derivations are not always the correct ones. By assigning weights to the rules based on their frequency in the input data, the quality of the results appears to have been improved. We have evaluated 20 graphs manually. When used UD graphs as inputs, the resulting graphs were correct, but when generating from raw text, we only managed to achieve an accuracy of 33% (regarding the 1st derivations). We plan to do further evaluation.

5 Conclusion

In this paper we described a program which is capable of automatic IRTG rule generation from parallel constituency tree and dependency graph data. Rules are generated based on several pre-defined parameterized templates. The resulting grammar implements a mapping between four formalisms, i.e. strings, the output of the Stanford parser, UD and 4lang, allowing conversion from any of these algebraic structures to any or all of the others. It covers 83% of the NPs of the Penn Treebank and it also has a limited capacity to generate 4lang graphs from dictionary definitions. We have assigned preliminary weights to the rules to create a probabilistic version of the grammar which improved the quality of the resulting derivations. If a single probabilistic IRTG were to implement the parallel parsing of strings, syntactic constituency structures, dependency graphs and semantic graphs, it could be trained simultaneously on each of these types of gold-standard data, resulting in a single end-to-end system for semantic parsing.

References

[1] A. Koller, "Semantic construction with graph grammars," in *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, (London), 2015.

[2] M.-C. De Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, and C. D. Manning, "Universal Stanford dependencies: A cross-linguistic typology.," in *LREC*, vol. 14, pp. 4585–92, 2014.

[3] A. Kornai, J. Ács, M. Makrai, D. M. Nemeskey, K. Pajkossy, and G. Recski, "Competence in lexical semantics," in *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM 2015)*, (Denver, Colorado), pp. 165–175, Association for Computational Linguistics, 2015.

[4] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," 1993.

[5] A. Koller and M. Kuhlmann, "A generalized view on parsing and translation," in *Proceedings of the 12th International Conference on Parsing Technologies*, pp. 2–13, Association for Computational Linguistics, 2011.

[6] B. Courcelle, "Graph grammars, monadic second-order logic and the theory of graph minors," *Contemporary Mathematics*, vol. 147, pp. 565–565, 1993.

[7] A. Koller and M. Kuhlmann, "Decomposing tag algorithms using simple algebraizations," in *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ 11)*, pp. 135–143, 2012.

[8] E. Ács, A. Holló-Szabó, and G. Recski, "Parsing noun phrases with Interpreted Regular Tree Grammars," in *XV. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2019)* (G. Berend, G. Gosztolya, and V. Vincze, eds.), pp. 301–313, 2019.

[9] M.-C. DeMarneffe, W. MacCartney, and C. Manning, "Generating typed dependency parses from phrase structure parses," in *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, vol. 6, (Genoa, Italy), pp. 449–454, 2006.

[10] G. Recski, "Building concept definitions from explanatory dictionaries," *International Journal of Lexicography*, 2018.