4Q1-5in2

Characterization of a Tree Mapping Algorithm for Tree-to-Tree Transducer Induction

Pascual Martínez-Gómez[1] Yusuke Miyao[1,2,3]

[1]Artificial Intelligence Research Center, AIST
[2]National Institute of Informatics and JST, PRESTO
[3]The Graduate University for Advanced Studies (SOKENDAI)

We characterize a tree mapping search space in terms of the tree fragment depth and number of variables, which are parameters of the resulting tree transducer grammar. We show how such characterization explains the trade-off between computational complexity and tree transducer expressivity. We evaluate our induced tree transducers on a Question-Answering task, quantifying accuracy and average tree mapping time as a function of our parameterization.

1. Introduction and Related Work

Tree-to-tree transducers are formal and well-studied models [Rounds, 1970, Thatcher, 1970] that describe the relationship between two trees (see [Knight and Graehl, 2005] for an overview). These models have been used to great success in multiple applications for natural language that require tree transformations, such as machine translation [Knight and Graehl, 2005], text summarization [Cohn and Lapata, 2009], question answering [Jones et al., 2012], paraphrasing and textual entailment [Wu, 2005]. However, inducing a tree transducer is difficult when the number of example tree pairs is small and the language variability is large.

[Martínez-Gómez and Miyao, 2016] designed a generalized tree mapping algorithm and a procedure to extract rules from pairs of trees, allowing the application of these models to a Question-Answering (QA) task over a large Knowledge Base. In this paper, we formally characterize the search space of the tree mapping algorithm and evaluate the expressivity of the resulting tree transducer in terms of QA accuracy. Our key result is a parameterization that allows to easily trade model expressivity by accuracy and that can be useful in other tree-to-tree transformation tasks.

2. Methodology

Following [Graehl and Knight, 2004], we define a tree transducer as a 5-tuple $(\mathcal{Q}, \Sigma, \Delta, q_{\text{start}}, \mathcal{R})$ where $\mathcal{Q}$ is the set of transducer states, Σ is the set of symbols of the input language (syntactic categories and English words), Δ is the set of symbols of the output language (KB entities and relations), q_{start} is the initial state, and $\mathcal{R}$ is the set of transducer rules. Transducer rules $r_i \in \mathcal{R}$ have the form $q.t_i \overset{w}{\to} t_o$, where $q \in \mathcal{Q}$ is the rule state, t_i is an input tree fragment, t_o is an output tree fragment, and w is the weight (or score) of the rule.

In our work, we commit to *extended*[*1] *root-to-frontier*[*2] *linear*[*3] transducers [Maletti et al., 2009], possibly with *deleting*[*4] operations. In this type of transducer, t_i and t_o in final rules are subtrees, whereas in non-final rules, t_i and t_o have variables in some leaves, specifying the connecting points with other tree fragments.

Let p be a path that uniquely identifies a node in a tree, equivalent to a Gorn address [Gorn, 1965]. For example, $p = (0)$ identifies the left-most child (child in branch index 0) of the root; $p = (0, 1)$ identifies the second child of the first child of the root; $p = ()$ identifies the root. Let $p_1 \cdot p_2$ be the concatenation of two paths, e.g. $p_1 = (a, b)$ and $p_2 = (c, d)$ would result in $p_1 \cdot p_2 = (a, b, c, d)$, and let the operator $|p|$ denote the length of a path, e.g. $|p_1| = 2$. We define a tree fragment $t \in \mathcal{T}$ as $s \downarrow p \perp \{q_1, \ldots, q_n\}$, a tree fragment from tree s rooted at p with n variables substituting subtrees at subpaths q_i for $1 \leq i \leq n$. The order of $\{q_1, \ldots, q_n\}$ matters, allowing to describe tree-to-tree relations with different branch orders. Note that p is a prefix of all q_i and q_i is not the prefix of any other subpath q_j for $i \neq j$. We can define the space of tree fragments of s rooted at any node p_i as:

$$\begin{aligned}\mathcal{T}^s_{p_i} = \{ & s \downarrow p_i \perp \{q_1, \ldots, q_n\} \mid \\ & q_i \in \mathcal{P}_s \wedge p_i \cdot r = q_i \wedge |r| \leq d, \\ & 1 \leq i \leq n\} \end{aligned} \quad (1)$$

where $\mathcal{P}_s$ is the set of all possible paths in tree s. The space $\mathcal{T}^s_{p_i}$ is parameterized by d, that limits the depth of tree fragments, thus imposing constraints on the exponential growth. The parameter n limits the number of variables, which limits the factorial combinations of branch re-orderings.

The complete space of tree fragment pairs is then $\mathcal{T}^s_{p_i} \times \mathcal{T}^t_{p_o}$ for all $p_i \in \mathcal{P}_i$ paths in source tree s and all $p_o \in \mathcal{P}_o$ paths in target tree t.

A naïve search algorithm that finds the minimum mapping cost between s at path p_i and t at path p_o would be given by the following recursive formula:

Contact: Pascual Martínez-Gómez, AIRC-AIST, AIST Tokyo Waterfront, 3-3-26, Aomi, Koto-ku, Tokyo, pascual.mg@aist.go.jp

*1 t_i may have depth larger than 1.
*2 Top-down transformations.
*3 t_i variables appear at most once in the t_o.
*4 Some variables on the t_i may not appear in the t_o.

$$\mathrm{C}\left(s \downarrow p_i, t \downarrow p_o\right) = \min_{\mathbf{q},\mathbf{q}'}\{\gamma\left(s \downarrow p_i \perp \mathbf{q}, t \downarrow p_o \perp \mathbf{q}'\right) + \sum_{j=1}^{|\mathbf{q}|} \mathrm{C}\left(s \downarrow q_j, t \downarrow q'_j\right)\} \quad (2)$$

where $s \downarrow p_i \perp \mathbf{q} \in \mathcal{T}^s_{p_i}$, $t \downarrow p_o \perp \mathbf{q}' \in \mathcal{T}^t_{p_o}$, $q_j \in \mathbf{q}$ and $q'_j \in \mathbf{q}'$. The cost $\gamma\left(t_i, t_o\right)$ is user defined and it usually depends on the downstream application. The mapping cost between s and t is finally given by the expression $\mathrm{C}\left(s \downarrow (), t \downarrow ()\right)$, where the node-to-node correspondences could be recovered by using back-pointers in a dynamic programming implementation. However, we do not actually implement such naïve algorithm, and we replace that search by a bottom-up beam-search algorithm (see [Martínez-Gómez and Miyao, 2016] for the details).

3. Experiments

In this section we evaluate the impact of different values of the tree fragment depth d and number of variables n on the accuracy of a downstream application. The application is Question-Answering over Freebase, a large Knowledge Graph with millions of facts. The problem is: given a natural language question, e.g. "*how many teams participate in the uefa*", transform its syntactic tree into a tree that represents its executable semantics, e.g.

```
(ID count (ID Team (ID League Uefa)))
```

This executable meaning representation can then be deterministically converted into a SPARQL query and triggered against Freebase to obtain the desired answer.

We evaluate on the FREE917, a corpus of 641 question-query pairs for training and 276 questions for testing. We extract rules when constraining the tree mapping search space by d and n, and then estimating the parameters of the resulting tree-to-tree transducer using the latent variable averaged structured-perceptron. At decoding, we generate $10,000$ target trees (executable meaning representations), convert them into SPARQL queries, and retain those that are both syntactically correct and retrieve more than zero results. We count a point of accuracy if the first tree (highest scoring tree) retrieves the correct answer, and we count a point of coverage if at least one tree in the $10,000$ target trees retrieves the correct answer.

Results are in Table 1. The system `t2t-d∞-n∞` stands for the tree-to-tree transducer induced with unrestricted depth and number of variables for the tree fragments t_i and t_o. The rest of the systems impose constraints on these parameters. For example, `t2t-d4-n∞` sets a limit of depth 4 for the tree fragments, but not limit for the number of variables it uses.

As we can observe, the accuracy drops but the tree mapping time decreases when we limit the depth of tree fragments and number of variables to $d \leq 2$ or $n = 1$. As the depth of tree fragments and number of variables increase, the accuracy saturates between .63 and .65. In these settings, the average number of rules per grammar decreases since the rules are larger (larger tree fragments). The average tree mapping time also increases proportionally to d and n, but we cannot observe the asymptotic trend possibly due to the small training dataset and the relatively small tree size of questions and semantic representations.

Systems	Acc.	Cov.	# Rules	Time
`t2t-d∞-n∞`	.64	.79	708	3.9, 1.7, 166.6, 8.6
`t2t-d1-n∞`	.36	.66	1958	1.3, 0.9, 16.5, 1.1
`t2t-d2-n∞`	.56	.84	886	1.7, 1.1, 24.3, 2.0
`t2t-d3-n∞`	.65	.80	746	2.4, 1.3, 45.7, 3.5
`t2t-d4-n∞`	.64	.79	713	2.8, 1.4, 67.0, 4.7
`t2t-d5-n∞`	.64	.79	708	3.0, 1.4, 87.1, 5.5
`t2t-d∞-n1`	.09	.30	1228	3.0, 2.0, 44.0, 3.3
`t2t-d∞-n2`	.63	.83	743	3.4, 1.9, 88.5, 5.3
`t2t-d∞-n3`	.63	.79	713	3.6, 1.8, 128.5, 6.9
`t2t-d∞-n4`	.63	.78	706	4.2, 1.9, 149.8, 8.6
`t2t-d∞-n5`	.63	.78	708	4.1, 1.9, 154.0, 8.3

Table 1: Accuracy and coverage results; average number of rules and tree mapping time (average, median, maximum and standard derivation) across all tree pairs.

4. Conclusion

In this paper we have showed a parameterization of the tree mapping search space that trades expressivity by computational complexity when inducing tree-to-tree transducers. We evaluated these tree transducers in terms of their semantic parsing accuracy when transforming the syntactic tree of a question into the tree of an executable meaning representation (a SPARQL query). We found that restricting the depth of the tree fragments to $d \leq 2$ or the number of variables to $n = 1$ had a negative impact on the model accuracy. Good results were obtained for $d = 3$ or $n = 2$, and no further significant improvements were obtained for $d \geq 4$ and $n \geq 3$, perhaps due to the typically small size of the questions and target trees in the FREE917 corpus.

The size of the transducer grammars in terms of number of rules decreased for $d > 1$ and $n > 1$, since the input and output tree fragments (t_i and t_o) of those rules are larger. The average tree mapping time increased proportionally to d and n, but no asymptotic trend can be observed given the large standard deviation.

Acknowledgments

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and is also supported by JSPS KAKENHI Grant Number 16K16111.

References

[Cohn and Lapata, 2009] Cohn, T. A. and Lapata, M. (2009). Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34:637–674.

[Gorn, 1965] Gorn, S. (1965). Explicit definitions and linguistic dominoes. In *Systems and Computer Science, Proceedings of the Conference held at Univ. of Western Ontario*, pages 77–115.

[Graehl and Knight, 2004] Graehl, J. and Knight, K. (2004). Training tree transducers. In Susan Dumais, D. M. and Roukos, S., editors, *HLT-NAACL 2004: Main Proceedings*, pages 105–112, Boston, Massachusetts, USA. Association for Computational Linguistics.

[Jones et al., 2012] Jones, B. K., Johnson, M., and Goldwater, S. (2012). Semantic parsing with bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 488–496, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Knight and Graehl, 2005] Knight, K. and Graehl, J. (2005). An overview of probabilistic tree transducers for natural language processing. In Gelbukh, A., editor, *Computational Linguistics and Intelligent Text Processing*, volume 3406 of *Lecture Notes in Computer Science*, pages 1–24. Springer Berlin Heidelberg.

[Maletti et al., 2009] Maletti, A., Graehl, J., Hopkins, M., and Knight, K. (2009). The power of extended top-down tree transducers. *SIAM Journal on Computing*, 39(2):410–430.

[Martínez-Gómez and Miyao, 2016] Martínez-Gómez, P. and Miyao, Y. (2016). Rule extraction for tree-to-tree transducers by cost minimization. In *Proc. of EMNLP*, pages 12–22.

[Rounds, 1970] Rounds, W. C. (1970). Mappings and grammars on trees. *Mathematical systems theory*, 4(3):257–287.

[Thatcher, 1970] Thatcher, J. W. (1970). Generalized sequential machine maps. *Journal of Computer and System Sciences*, 4(4):339 – 367.

[Wu, 2005] Wu, D. (2005). Recognizing paraphrases and textual entailment using inversion transduction grammars. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, EMSEE '05, pages 25–30, Stroudsburg, PA, USA. Association for Computational Linguistics.