Controlling Expressiveness of Question Interpretation with a Constrained Tree Transducer Induction

Pascual Martínez-Gómez
pascual.mg@aist.go.jp
Artificial Intelligence Research Center, AIST
Tokyo, Japan

Yusuke Miyao*†
yusuke@nii.ac.jp
National Institute of Informatics
Tokyo, Japan

ABSTRACT

An important problem in Question Answering over Knowledge Bases is to *interpret a question* into a database query. This problem can be formulated as an instance of *semantic parsing* where a natural language utterance is analyzed into a (possibly executable) meaning representation. Most semantic parsing strategies for Question Answering use models with limited expressiveness because it is difficult to characterize it and systematically control it. In this work we use tree-to-tree transducers which are very general and solid models to transform the syntactic tree of a question into the executable semantic tree of a database query. When designing these tree transducers, we identify two parameters that influence the construction cost and their expressive capabilities, namely the tree fragment depth and number of variables of the rules. We characterize the search space of tree transducer construction in terms of these parameters and show considerable improvements in accuracy as we increase the expressive power.

CCS CONCEPTS

• **Information systems** → **Question answering**; • **Theory of computation** → *Tree languages*; • **Computing methodologies** → *Natural language processing*;

KEYWORDS

Question Answering, Tree Transducers

ACM Reference format:
Pascual Martínez-Gómez and Yusuke Miyao. 2017. Controlling Expressiveness of Question Interpretation with a Constrained Tree Transducer Induction. In *Proceedings of ACM SIGIR conference, Tokyo, Japan, September 2017 (OKBQA'17),* 5 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION AND RELATED WORK

Question Answering (QA) over Knowledge Bases (KBs) is a step forward in the realization of human-machine natural language interfaces to large structured knowledge resources. In this task, one of the main challenges is the interpretation of a natural language utterance into an executable meaning representation. In the community of Natural Language Processing, this task can be formulated as a semantic parsing problem where the objective is to produce a symbolic meaning representation with predicates grounded to Knowledge Base constants (entities and relations). This problem is different from that of the *Simple Questions* tasks [2, 21] where researchers try to identify a single fact from a KB given a short question that typically involves only one relation. Instead, we aim to answer questions that require higher levels of compositionality, aggregation or KB inference.

There are two main strategies when doing executable semantic parsing for QA over large KBs. The first one is that of Berant et al. [1] where a question is directly parsed into a semantic formula without an intermediate syntactic representation (string-to-tree transformations). In this approach, the grammar of the executable semantic representation is manually specified[1] and the parameters of a statistical model are estimated with the objective to guide the parser towards correct derivations. The second strategy follows the principles of the syntax-semantic interface, which is a popular paradigm of semantic compositionality among linguists and formal semanticists (tree-to-tree transformations). In this strategy, the syntactic analysis of a question (or more generally, a sentence) is used to guide the semantic composition of a symbolic meaning representation. Some early representatives are the work of Ge and Mooney [5] and that of Wong and Mooney [19]. Most recently, dependency trees of questions are transformed into grounded meaning representations (SPARQL queries) using a set of manually designed rules [14], establishing a new state of the art.

We commit to this second strategy with tree-to-tree transducers which are general and well-studied models [15, 17] that describe how input trees can be transformed into output trees. Knight and Graehl [10] give a good overview. Given the generality of these models, they have been used in a variety of text transformation tasks such as paraphrasing and textual entailment [20], text summarization [4] or question answering [8]. However, it is difficult to induce these tree transducers in semantic parsing tasks where there is a large vocabulary in the target language (i.e. number of constants in the KB) and typically small numbers of examples of tree pairs in the training set. Martínez-Gómez and Miyao [13] proposed a tree mapping algorithm that served as the basis to induce tree transducers from small data, allowing the application of these models to Question Answering tasks over large Knowledge Bases. However, they did not study how transducer rules with different

*Also with Artificial Intelligence Research Center, AIST, Tokyo, Japan.
†Also with The Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan.

OKBQA'17, September 2017, Tokyo, Japan

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

[1] These grammars are typically small. Thus, manual specification is often feasible.

expressiveness affect model accuracy in a downstream application and its impact in the transducer construction cost.

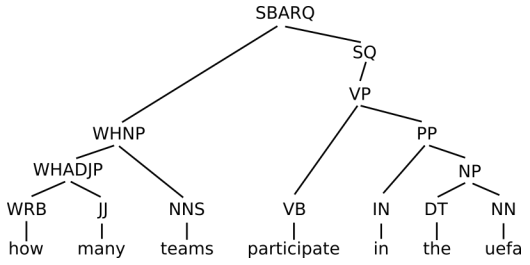
Our contribution is a formal characterization of the search space in the tree mapping algorithm that induces tree transducer grammars. This characterization evidences two critical parameters that control the model expressiveness and its complexity, which we believe is useful in text transformation tasks that deploy synchronous tree grammars. We evaluate the expressiveness of the resulting tree transducers in terms of QA accuracy and we demonstrate the importance of tree-to-tree transformation models whose rules consume and produce tree fragments of depth larger than one.

2 BACKGROUND

Given a question and a Knowledge Base, our system performs the following steps. First, obtain the constituent syntactic tree of the question. Second, use a set of weighted rules to transform fragments of the syntactic tree into fragments of a semantic tree and compose the executable meaning representation. Finally, execute the meaning representation (i.e. SPARQL query) on a KB and return the results. As a running example, consider the question:

Q: *how many teams participate in the uefa*

which is syntactically analyzed into the following constituent tree:



(1)

The objective is to transform such a syntactic tree into the following SPARQL query:

```
SELECT COUNT(?x) WHERE {
  ?a  Team    ?x    .
  ?a  League  Uefa  . }
```

(2)

which corresponds to the following λ expression:

$$\texttt{count}(\lambda x.\exists a.\texttt{Team}(x, a) \wedge \texttt{League}(a, \texttt{Uefa})) \tag{3}$$

where `Team` and `Uefa` are the KB entities to which the natural language expressions *teams* and *uefa* map into. Note that the expressions in Equation 2 and Equation 3 do not have a tree structure but a graph structure due to the presence of repeated variables (?a or a) at the leaves. However, it is convenient to shape these graph structures into the form of a tree. To this end, Liang [11] proposes the λ-DCS tree language, where existentially quantified variables are made implicit. For our running example, the λ-DCS expression would be:

$$\texttt{count(Team.League.Uefa)} \tag{4}$$

which can be trivially represented as a semantic tree structure:

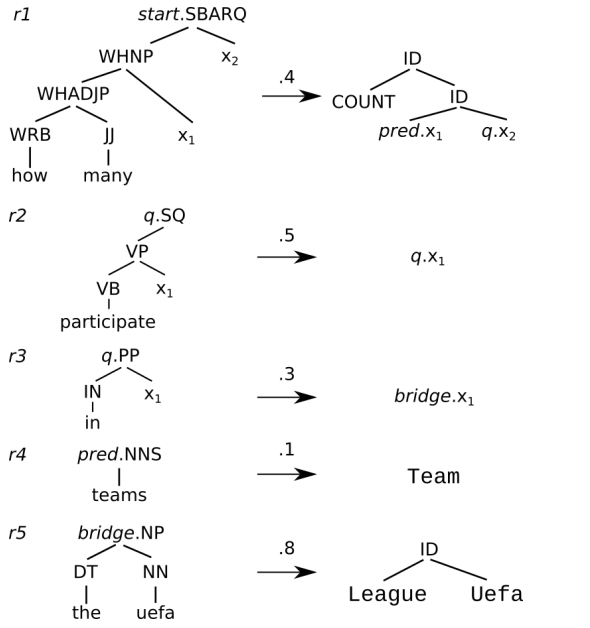
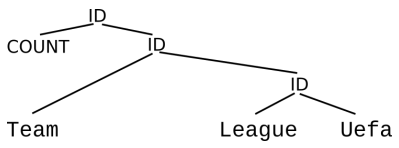


Figure 1: Transducer rules that transform syntactic tree (1) into executable semantic tree (5).



(5)

Thus, we transform the syntactic tree s in (1)[2] into the executable semantic tree t in (5)[3] which can be later trivially converted into the SPARQL query in (2).

The tree-to-tree transformation from (1) to (5) can be performed with a set of weighted rules (see Figure 1) whose left-hand-sides match and consume fragments of the syntactic tree (1) and produce tree fragments of the executable semantic tree (5). These rules are at the core of a tree transducer, which we describe now.

Following the same terminology as Graehl and Knight [7], a tree transducer is a 5-tuple $(Q, \Sigma, \Delta, q_{\text{start}}, \mathcal{R})$ where Q is the set of transducer states that carry some memory through the transformation process, Σ is the set of input symbols (i.e. syntactic categories and English words), Δ is the set of output symbols (KB entities and relations), q_{start} is the initial state from which the tree transformation starts, and $\mathcal{R}$ is the set of transducer rules. Transducer rules $r_i \in \mathcal{R}$ define atomic transformations and they have the form $q.t_i \xrightarrow{w} t_o$, where $q \in Q$ is the rule state, t_i is an input (syntactic) tree fragment, t_o is an output (semantic) tree fragment, and w is the weight (or score) of the rule. In our work, we commit to *extended*[4]

[2] We also call it *source* tree s.
[3] We also call it *target* tree t.
[4] t_i may have depth larger than 1.

root-to-frontier[5] *linear*[6] transducers [12], possibly with *deleting*[7] operations. Some rules are *terminal rules* whose t_i match entire syntactic subtrees and whose t_o produce semantic subtrees (e.g. $r4$ and $r5$). Other rules (e.g. $r1 - r3$) are *non-terminal* rules where variables x_i are connection points with other rules thus carrying over the tree compositionality.

Note in Figure 1 how some rules are more complex (and expressive) than others. For instance, the left-hand-side of $r4$ is a syntactic subtree of depth 1 (one-level subtree) with no variables, whereas the left-hand-side of $r1$ has depth 4 and two variables. We formalize this concept in the next section and characterize the space of possible rules by parameterizing it in terms of rule depth and number of variables.

3 METHODOLOGY

In our characterization, the expressiveness of a tree transducer depends on the expressiveness of its rules since we keep the states Q and input/output vocabulary (Σ and Δ) constant. In turn, the expressiveness of the transducer rules ($r = q.t_i \xrightarrow{w} t_o$) depends on the characteristics of the input and output tree fragments (t_i and t_o). Thus, if we want to characterize and parameterize the space of induced transducers we need to characterize the space of possible tree fragments. To this end, we need to introduce terminology that allows us to define tree fragments with precision.

We uniquely identify nodes in a tree by using paths $p \in \mathcal{P}$, which are similar to Gorn addresses [6] but with a tuple notation. For example, in tree (1), the path $p = (0)$ identifies the node with syntactic category `WHNP`, that is, the child index 0; the path $p = (0, 1)$ identifies the node `NNS` and $p = ()$ identifies the root. For convenience we define the path concatenation operation as $p_1 \cdot p_2$. For example, given $p_1 = (a, b)$ and $p_2 = (c, d)$, their concatenation results in $p_1 \cdot p_2 = (a, b, c, d)$ with a path length $|(a, b, c, d)| = 4$.

We define a tree fragment $t \in \mathcal{T}$ as $s \downarrow p \perp \{q_1, \ldots, q_n\}$, which is a tree fragment from tree s rooted at path $p \in \mathcal{P}$ with n variables substituting subtrees at subpaths $q_i \in \mathcal{P}$ for $1 \leq i \leq n$. Note that different orders of $\{q_1, \ldots, q_n\}$ allow to describe tree transformations that swap branches. The path p is a prefix of all q_i and q_i is not the prefix of any other subpath q_j for $i \neq j$ since variables can only appear at the leaves of tree fragments (no variable can have children). In our running example, the right-hand-side of $r1$ would be a tree fragment $t \downarrow () \perp \{(1, 0), (1, 1)\}$ where t is the target (semantic) tree, $p = ()$ since the rule starts at the root of t and there are two subpaths $q_1 = (1, 0)$ and $q_2 = (1, 1)$ that specify the location of each of the two variables x_1 and x_2. Another example would be the left-hand-side of $r4$, described as $s \downarrow (0, 1) \perp \{\}$ (note that there are no variables).

We can now define the space of tree fragments of a tree s rooted at any node p_i as:

$$\mathcal{T}_{p_i}^{s} = \{s \downarrow p_i \perp \{q_1, \ldots, q_n\} \mid q_i \in \mathcal{P}_s \wedge p_i \cdot r = q_i \wedge |r| \leq d, \; 1 \leq i \leq n\} \quad (6)$$

where $\mathcal{P}_s$ is the set of paths to all nodes in tree s. The space of tree fragments $\mathcal{T}_{p_i}^{s}$ is parameterized by i) the maximum depth d of tree fragments ($|r| \leq d$) which controls the exponential growth and ii) the maximum number of variables n which limits the factorial combinations ($n!$) of branch orderings.

The space of transducer rules[8] is formed by all pairs of tree fragments $\mathcal{T}_{p_i}^{s} \times \mathcal{T}_{p_o}^{t}$ for all $p_i \in \mathcal{P}_i$ paths in the syntactic source tree s and all $p_o \in \mathcal{P}_o$ paths in the semantic target tree t. The mapping cost between trees s and t at paths p_i and p_o can be computed by the following recursive formula:

$$\mathrm{C}\,(s \downarrow p_i, t \downarrow p_o) = \min_{\mathbf{q}, \mathbf{q}'} \{\gamma\,(s \downarrow p_i \perp \mathbf{q}, t \downarrow p_o \perp \mathbf{q}') + \sum_{j=1}^{|\mathbf{q}|} \mathrm{C}\left(s \downarrow q_j, t \downarrow q_j'\right)\} \quad (7)$$

where $s \downarrow p_i \perp \mathbf{q} \in \mathcal{T}_{p_i}^{s}$, $t \downarrow p_o \perp \mathbf{q}' \in \mathcal{T}_{p_o}^{t}$, $q_j \in \mathbf{q}$ and $q_j' \in \mathbf{q}'$. The cost between two tree fragments $\gamma\,(t_i, t_o)$ depends on the application. In our case, it is an ensemble of cost functions that assigns low costs to pairs of syntactic and semantic subtrees whose leaves (natural language phrases and KB constants) may have a linking relation.

The mapping cost between the roots of s and t can be computed as $\mathrm{C}\,(s \downarrow (), t \downarrow ())$ whereas the node-to-node correspondences can be recovered using back-pointers as it is usual in dynamic programming. For the sake of efficiency we perform the search using an approximate bottom-up beam-search algorithm [13][9] parameterized by d and n.

4 EXPERIMENTS

We use tree transducers to transform the syntactic tree of a question into a SPARQL query. We evaluate on Free917, a corpus of 641 question-query pairs for training and 276 questions for testing. We obtain syntactic constituent trees of questions using the Stanford caseless models [9] which produce trees with an average of 24.5 nodes and tree height 7.4 in this dataset. The gold queries in this dataset typically have between one and three statements, possibly with a *count* aggregator. We use the entity lexicon released by Cai and Yates [3] and the relation lexicon released by Martínez-Gómez and Miyao [13]. Our KB is the same Freebase dump as in Berant et al. [1], which contains millions of facts.

We automatically induce (construct) tree-to-tree transducers by extracting rules from pairs of question syntactic trees and query trees. This rule extraction is performed by a tree mapping search algorithm that is constrained by d and n, thus resulting in tree transducers with different levels of expressiveness[10]. The weights (parameters) of the resulting tree-to-tree transducers are estimated using the latent variable averaged structured perceptron. In this parameter estimation routine, rules are represented by a feature

[5] Top-down transformations.
[6] t_i variables appear at most once in the t_o.
[7] Some variables on the t_i may not appear in the t_o.

[8] If we ignore the rule states.
[9] https://github.com/pasmargo/t2t-qa
[10] That is, transducers with rules with a maximum of d tree fragment depth and a maximum of n variables.

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 for the test split of Free917. "# Rules" and "Time" stand for the average number of rules and tree mapping time (average, median, maximum and standard deviation) across tree pairs in the training set.

vector[11] and the rule score is the result of a linear combination between these rule features and model weights. We reward[12] weights associated to rule sequences that transform a question syntactic tree into a query that retrieves the correct answer as given in the gold training data. We perform 3 iterations over the whole training set and use the learned weights for the decoding stage. The number of iterations and the learning rate are estimated on a validation split from the training data.

For questions in the test set, our decoder generates 10,000 target trees which we trivially convert into SPARQL queries that we run against the KB. Then, we keep those that retrieve at least one answer. We count a point of accuracy if the highest scoring SPARQL query retrieves the correct set of answers whereas we count a point of coverage if at least one query in the 10,000 candidates retrieves the correct set of answers. Then we average the accuracy and coverage over the whole test set.

Our results in terms of accuracy and coverage for different settings of d and n are in Table 1. The table also displays the average number of rules extracted per tree pair in the training data and the average time, median, maximum and standard deviation of the tree mapping and rule extraction across all training tree pairs. The system t2t-d∞-n∞ imposes no constraints on d and n whereas the rest of the systems do. For example, the system t2t-d3-n∞ limits the tree fragment depth to $d \leq 3$ but imposes no constraints on the number of variables ($n \leq \infty$).

When inducing transducers with simple rules (depth $d \leq 2$ or $n = 1$), the accuracy and coverage is low but the tree mapping is fast. The accuracy (and coverage) increases progressively and saturates between .63 and .65 (.78 and .84) as we increase the rule expressiveness by setting higher limits in tree fragment depth and number of variables. However, tree mapping time also seems to increase proportionally to rule expressiveness in terms of d and n but no asymptotic trend can be observed due to the small training set and relatively small question complexity. The average number of rules changes only slightly for $d \geq 4$ and $n \geq 3$, which suggests that questions in Free917 do not require transformation operations more expressive than that.

As a comparison to other systems, SEMPRE [1] obtains and accuracy of .62, whereas Reddy et al. [14]'s DepLambda system obtains an accuracy of .78 and a coverage of .96. However, these systems are not directly comparable because they use different entity/relation linkers, manually specified grammars and (or) handcrafted rules.

5 FUTURE WORK AND CONCLUSION

We have concentrated on describing the search space of the question interpretation problem but we have neglected the grounding problem (mapping natural language expressions to KB constants) by re-using manually created lexicons of entities and relations. However, in the realization of a full-fledged QA system we need to integrate wide-coverage entity and relation linkers which we plan to do in the near future. Moreover, we claimed that tree transducers are general models but we evaluated on a single dataset. Our next step is to use other datasets such as WebQuestions [1], GraphQuestions [16] and QALD challenges [18] to assess the generality of these models. Yet another extension is to use these tree-to-tree transducers as an effective generalization of symbolic semantic parsing with grounding. This extension would encompass the current state of the art based on manually designed transformation rules for dependency trees but with the added advantage of including a bootstrapping mechanism to acquire new rules for questions with previously unseen syntactic structures.

We showed a characterization of the tree mapping search space that is parameterized by the tree fragment depth d and number of variables n in rules. Experimental results showed how these two parameters trade QA accuracy by tree mapping time. Specifically, we found that tree transducers whose rules are limited to $d \leq 2$ or $n = 1$ obtain a low accuracy but the tree mapping time to induce these transducers is fast. Higher accuracies were obtained when using more expressive rules ($d = 3$ or $n = 2$). However, no further gains were obtained for larger tree fragment depths and number of variables because the questions and target trees in the Free917 corpus are relatively simple and do not require induced rules with such a high level of expressiveness. As a comparison, SEMPRE and DepLambda have grammar rules of depth $d = 1$. The results in this paper suggest that those systems could also be improved by increasing their rule depth. However, since their grammars are hand-crafted, the manual specification of these complex rules is not trivial and requires very fine grained linguistic analysis.

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. We thank Yoshimasa Tsuruoka and the anonymous reviewers for their helpful comments.

[11] These features are instantiated using hand-engineered feature templates that measure rule characteristics such as the number of nodes in the left- or right-hand-side tree fragments, the presence of an aggregator function, n-gram overlap between words in the question and text literals associated to KB entities or relations, etc.

[12] That is, we increase their value by a small factor which is the learning rate: 0.01.

REFERENCES

[1] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, 1533–1544. http://www.aclweb.org/anthology/D13-1160

[2] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale Simple Question Answering with Memory Networks. *CoRR* abs/1506.02075 (2015). http://arxiv.org/abs/1506.02075

[3] Qingqing Cai and Alexander Yates. 2013. Large-scale Semantic Parsing via Schema Matching and Lexicon Extension. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 423–433. http://www.aclweb.org/anthology/P13-1042

[4] Trevor Anthony Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research* 34 (2009), 637–674.

[5] Ruifang Ge and Raymond J. Mooney. 2005. A Statistical Semantic Parser That Integrates Syntax and Semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CONLL '05)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 9–16. http://dl.acm.org/citation.cfm?id=1706543.1706546

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

[7] Jonathan Graehl and Kevin Knight. 2004. Training Tree Transducers. In *HLT-NAACL 2004: Main Proceedings*, Daniel Marcu Susan Dumais and Salim Roukos (Eds.). Association for Computational Linguistics, Boston, Massachusetts, USA, 105–112.

[8] Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 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)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 488–496. http://dl.acm.org/citation.cfm?id=2390524.2390593

[9] Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Sapporo, Japan, 423–430. https://doi.org/10.3115/1075096.1075150

[10] Kevin Knight and Jonathan Graehl. 2005. An Overview of Probabilistic Tree Transducers for Natural Language Processing. In *Computational Linguistics and Intelligent Text Processing*, Alexander Gelbukh (Ed.). Lecture Notes in Computer Science, Vol. 3406. Springer Berlin Heidelberg, 1–24. https://doi.org/10.1007/978-3-540-30586-6_1

[11] Percy Liang. 2013. Lambda Dependency-Based Compositional Semantics. *CoRR* abs/1309.4408 (2013). http://arxiv.org/abs/1309.4408

[12] Andreas Maletti, Jonathan Graehl, Mark Hopkins, and Kevin Knight. 2009. The Power of Extended Top-Down Tree Transducers. *SIAM J. Comput.* 39, 2 (2009), 410–430. https://doi.org/10.1137/070699160

[13] Pascual Martínez-Gómez and Yusuke Miyao. 2016. Rule Extraction for Tree-to-Tree Transducers by Cost Minimization. In *Proc. of EMNLP*. 12–22.

[14] Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the ACL* 4 (2016), 127–140.

[15] William C. Rounds. 1970. Mappings and grammars on trees. *Mathematical systems theory* 4, 3 (1970), 257–287. https://doi.org/10.1007/BF01695769

[16] Yu Su, Huan Sun, Brian Sadler, Mudhakar Srivatsa, Izzeddin Gur, Zenghui Yan, and Xifeng Yan. 2016. On Generating Characteristic-rich Question Sets for QA Evaluation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, 562–572. https://aclweb.org/anthology/D16-1054

[17] James W. Thatcher. 1970. Generalized sequential machine maps. *J. Comput. System Sci.* 4, 4 (1970), 339 – 367. https://doi.org/10.1016/S0022-0000(70)80017-4

[18] Christina Unger, Corina Forascu, Vanessa Lopez, Axel-Cyrille Ngonga Ngomo, Elena Cabrio, Philipp Cimiano, and Sebastian Walter. 2015. Question Answering over Linked Data (QALD-5). In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum*, Linda Cappellato, Nicola Ferro, Gareth Jones, and Eric San Juan (Eds.), Vol. 1391. Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum.

[19] Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for Semantic Parsing with Statistical Machine Translation. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL '06)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 439–446. https://doi.org/10.3115/1220835.1220891

[20] Dekai Wu. 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)*. Association for Computational Linguistics, Stroudsburg, PA, USA, 25–30. http://dl.acm.org/citation.cfm?id=1631862.1631867

[21] Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple Question Answering by Attentive Convolutional Neural Network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. The COLING 2016 Organizing Committee, Osaka, Japan, 1746–1756. http://aclweb.org/anthology/C16-1164