A Survey on Semantic Parsing for Machine Programming

CELINE LEE, JUSTIN GOTTSCHLICH, and DAN ROTH

Over the last decade, the fields of natural language processing and understanding have seen major advances. Yet, while some of these techniques can be applied to programming language processing and understanding, there remain some fundamental differences. In this survey paper, we attempt to provide an overview of the growing body of research in this space. We begin by reviewing natural language semantic parsing techniques and extracting lessons from the evolution of semantic parsing. We then draw parallels between modern efforts in neural semantic parsing and program synthesis. In closing, we present what we believe are some of the emerging open challenges in this domain.

Additional Key Words and Phrases: semantic parsing, code generation, program synthesis

1 INTRODUCTION

Machine programming (MP) is the field concerned with the automation of all aspects of software development [27]. According to Gottschlich et al. [27], the field can be reasoned about across three pillars: *intention*, *invention*, and *adaptation*. *Intention* is concerned with capturing user intent, whether by natural language, visual diagram, demonstration, or other techniques. *Invention* explores ways to construct higher-order programs through the composition of existing – or novel creation of – algorithms and data structures. *Adaptation* focuses on transforming higher-order program representations or legacy software programs to achieve certain characteristics (e.g., performance, security, correctness, etc.) for the desired software and hardware ecosystem.

In this paper, we discuss *(i)* semantic parsing and *(ii)* code generation, which chiefly fall within the boundaries of the intention and invention pillars, respectively. In the context of natural language (NL), semantic parsing is generally concerned with converting NL *utterances* (i.e., the smallest unit of speech) to structured logical forms. Once done, these logical forms can then be utilized to perform various tasks such as question answering [40, 41, 47, 76, 82], machine translation [5, 78], or code generation [37, 48, 69, 83], amongst other things. In the next Section 2, we provide a high-level overview of the evolution of techniques in NL semantic parsing. We explore the question of supervision for semantic parsing tasks in Section 3, then discuss modern advances in neural semantic parsing for code generation in Section 4. To conclude, we consider some potential future directions for semantic parsing for machine programming.

2 EVOLUTION OF SEMANTIC PARSING

This section will review the evolution of NL semantic parsing techniques from early hand crafted rules to the rise of learned grammars via statistical learning methods, to modern neural semantic parsing efforts. For a deeper analysis of the components and processes for NL semantic parsing and program synthesis, see Sections A and B in the Appendix.

Many early approaches to NL semantic parsing used hand-crafted syntactic pattern and phrase matching to parse the meaning from a given natural language input [38, 79]. Subsequent works [31, 49, 76] then augmented these rules with grammars that incorporate knowledge of semantic categories so that similar words can be generalized to sets of defined categories, which then constrain the output space of the logical forms. However, as a function of being based off of pattern-matching rules, these systems are often brittle, and as a function of relying on defined semantic categories, these grammar-constrained systems are frequently domain-specific.

Authors' address: Celine Lee; Justin Gottschlich; Dan Roth.

Statistical methods for the induction of NL semantic parsing grammars rose in the late 1990s and early 2000s, as researchers began to present systems that employ inductive logic programming methods to *learn* parsers and transformation rules to map natural language inputs to semantic parsing meaning representations [40, 85]. Other probabilistic learning approaches induced grammars to represent the underlying semantics of a knowledge base. These grammars could then serve as a powerful tool to capture textual semantics by mapping natural language sentences to their logical forms [8, 9, 86, 87]. However, a prominent challenge in inducing or learning these rules is scalability: how to obtain labeled data from which to learn statistical inference, and how to scale the probabilistic models to more complex domains. We refer the reader to Appendix Section C.1 for a more in-depth discussion of statistical models to induce grammars for semantic parsing.

The modern rise in neural models has been marked by the increase in papers exploring encoder-decoder sequence-to-sequence (seq2seq) frameworks for semantic parsing [21]. These encoders and decoders are often built with recurrent architectures such as long short-term memory (LSTM) cells in an effort to better capture the dynamic and sequential nature of both natural and programming languages. Neural models can enable systems to automatically induce grammars, templates, and features instead of relying on humans to manually annotate them. This can provide the model with flexibility to generalize across languages and representations [41]. The seq2seq structure may resemble machine translation systems [78], which traditionally translate a natural language input into another unstructured natural language output [5]; however, NL semantic parsing systems produce a *structured* logical form. The formal output structure of semantic parsing enables neural decoders to lean on the logic of the induced grammars, while retaining the simplicity of neural translation without latent syntactic variables [41, 64, 83]. Likewise, due to demand for proper construction of the formal output structure, these learned systems can have more rigid requirements for acceptable outputs. Progress made using neural models in the space of NL semantic parsing suggests that neural models may also be a powerful tool for code generation due to their expressivity and adaptability. It should be noted, however, that these neural models tend to have a weakened ability to leverage the logical compositionality and thus interpretability of traditional rules-based approaches. These neural NL semantic parsing systems additionally face the challenge of rare data points– it can be difficult to estimate reliable parameters or induce operations for input predicates rarely seen in the training data. A more in-depth discussion of modern neural semantic parsing techniques is presented in Section 4, where we focus on semantic parsing models for code generation.

3 SUPERVISION IN SEMANTIC PARSING

Parallel to the evolution of NL semantic parsing models is the evolution of supervision for semantic parsing. Fully-supervised learning customarily mandates a corpus of paired natural language utterances and their corresponding logical forms. However, this can be expensive to generate and difficult to scale. Alternative forms of supervision for semantic parsing have included supervision by denotation, weakly-supervised learning, and self-supervised learning.

In supervision by denotation, instead of directly evaluating the success of a semantic parser by comparing its output to a gold standard "correct" logical form as done in fully-supervised learning, learning can be driven by the response of the context or environment [19, 47]. However, at least two primary challenges arise with supervision by denotation. *(1)* The space of potential denotations may be large. This can make the search problem for intermediate latent representations computationally intractable. *(2)* This weaker response signal has the potential for spurious logical forms: semantic parsing outputs that happen to produce the correct response in the environment, but do not actually carry the correct semantics. Modifications to supervision by denotation have been proposed to address these challenges.

Weakly-supervised learning and self-supervised learning are presented as alternative, more scalable approaches to semantic parsing. Experiments in using conversational feedback [8] and execution in a robotics domain [9], among others, have highlighted the feasibility of using weaker supervision signals to obtain competitive semantic parsing results. Self supervision techniques have also been devised to iteratively re-train the system on high-confidence datapoints [25], induce a semantic grammar over an input space [61], and recursively cluster semantically-equivalent logical forms [62]. These techniques only need corpora of unlabeled input data in some domain, and proceed to infer the rest: often the structure of the semantic space and how to parse NL inputs to logical forms in that space. Recent work in open vocabulary NL semantic parsing systems replace a formal knowledge base with a probabilistic database learned from the text corpus [24, 42, 46]. The reliance of weak and self supervision on data over expert rules or annotation seems to have great potential for future machine learning applications as the field moves into more deep learning models, but current approaches may face such challenges as overfitting, weaker evaluation, and longer training times.

See Appendix Section D for a more in-depth discussion of techniques used for supervision by denotation, weakly-supervised learning, and self-supervised learning.

4 MODERN ADVANCES IN NEURAL SEMANTIC PARSING FOR CODE GENERATION

Recent semantic parsing efforts have largely centered around neural models, especially encoder-decoder networks. In this section, we provide a short summary of modern advances in neural models for semantic parsing of natural language utterances to code generation. For more depth into the topic, we refer the reader to Appendix Section E.

Information to generate a NL semantic parse may not always be encapsulated entirely in a single input sentence. Surrounding context can provide valuable information for disambiguation in NL semantic parsing [8]. Context can be fed into the encoder to incorporate into encoder-decoder models [37]. A supervised copy mechanism [28] based on attention weights can also be used to copy an environment token to the output, allowing a model to copy in variable names not seen in the training data or in the input utterance.

Grammar also can guide and constrain the decoding process of neural encoder-decoder models. Incorporating grammatical constraints into the decoder derivation sequence [41, 80] has been demonstrated to bolster accuracy. This technique additionally has the benefit of including explicit entity linking during encoding to ensure the generation of well-typed logical forms. Additionally, code idioms, bits of code that occur frequently across a code repository and tend to have a specific semantic purpose, can be used to enhance grammars in semantic parsing [35, 69].

Solar-Lezama et al. [71] present the concept of sketching to generate programs from a high-level description. Through sketching, developers communicate insight through a partial program, i.e. the sketch, that describes the high-level structure of an implementation. Synthesis procedures can then fill in the holes of the sketch to generate the full implementation for a given programming language. In NL semantic parsing, encoder-decoder systems have been presented that encode natural language inputs then decode at varying levels of abstraction: first to create a sketch, then to fill in its missing low-level details [22, 56].

A remaining challenge in neural semantic parsers is interpretability. Uncertainty modeling [23] for semantic parsing models allows models to better understand what it does and does not know. The ability to target weak parts of a model leads us to conversational programming approaches, where the model might query the user for missing or uncertain information. By engaging in a back-and-forth dialogue between the program and the human user, gaps and errors in logical forms can be filled and remedied, respectively. Additionally, clarifications can reduce the search space to promise a more accurate output [8, 16, 51].

5 FUTURE DIRECTIONS

The task of connecting human communication with machine data and capabilities, as in the intention pillar of machine programming, has seen considerable headway over the past few decades. Moving forward in semantic parsing for code generation, we identify three areas of focus: model evaluation, semantics representations, and data.

Model Evaluation. Current methods of evaluation for semantic parsing correctness use strict matches to gold standards, denotation, hand-wavy heuristics, or some other imperfect method of measuring understanding of semantic meaning. Exact match evaluation may incorrectly penalize mismatching but semantically accurate logical forms. Supervision by denotation can incorrectly reward spurious or inefficient nonsensical logical forms. Hand-wavy heuristics such as Bleu score [59], which evaluates the "quality" of machine translation outputs via a modified n-gram precision metric, have been shown to not reliably capture semantic equivalence for natural language [14] or for code [32, 75].

Model evaluation can be particularly difficult with neural models, which tend to function more as "black boxes" whose induced logic is mostly evaluated by examining the outputs. We recommend future research to consider how to design neural models to better capture the complex nature of both natural language and source code in translating natural language into source code. Compositional generalization may be one way to evaluate how well a model understands natural language. Oren et al. [58] investigate compositional generalization in semantic parsing. They find that while well-known and novel extensions to the seq2seq model improve performance, performance on in-distribution data is still significantly higher than on out-of-distribution data. This suggests a need to make further changes to the neural encoder-decoder approach in order to inject fundamental "understanding." Neural Module Networks (NMNs) can provide interpretability, compositionality, and improved generalizability to semantic parsers [30, 30, 74]. However, the advanced interpretability of NMNs comes at the cost of restricted expressivity because each NMN needs to be individually defined. Wong et al. [77]'s system for inductive program synthesis guided by natural language annotations operates on the intuition that natural language can offer some ideas about the structure of a program search space. It extracts words and NL primitives from language annotations to learn program primitives that "boostrap" future learning and enable compositional generalization in synthesizing programs.

Code Semantics Representation. We perceive that an ongoing challenge in semantic reasoning about language and code may be the lack of a unified code representation that can capture semantics and details enough to resolve ambiguity but remain agnostic of programming language. The graph-based meaning representations discussed in Appendix Section A.3 are promising avenues to explore. They have shown to be effective in code retrieval, so we propose they be deployed in a semantic parsing framework to evaluate their viability for semantic parsing and code generation.

Data. As the field of semantic parsing for code generation starts working with larger programs and general-purpose languages, existing datasets that match natural language utterances to code snippets will likely not be enough. To obtain more data for neural deep learning, we recommend using data generation models such as Snorkel [66]. Snorkel employs weak supervision to create training data without any need for hand-labeled data. We also suggest that more datasets be developed that account for longer chunks of code, such as the APPS dataset [32]. This can be done by factoring in context variables and functions, or by mining larger chunks of code with periodic natural language annotations in the form of comments. There may be room for further research in natural language understanding to design models that intuit programming concepts from plain English, free of such technical jargon as *"list," "graph,"* and *"iterate,"*. HEXAGONS [45] is one such example of a work that takes a step toward understanding high-level abstractions and more sophisticated communication features in natural language.

REFERENCES

[1] Omri Abend and Ari Rappoport. 2013. Universal Conceptual Cognitive Annotation (UCCA). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 228–238. https://www.aclweb.org/anthology/P13-1023

[2] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. 2018. Learning to Represent Programs with Graphs. In *International Conference on Learning Representations*.

[3] Uri Alon, Omer Levy, and Eran Yahav. 2019. code2seq: Generating Sequences from Structured Representations of Code. In *International Conference on Learning Representations*.

[4] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. 2019. Code2vec: Learning Distributed Representations of Code. *Proc. ACM Program. Lang.* 3, POPL, Article 40 (Jan. 2019), 29 pages. https://doi.org/10.1145/3290353

[5] Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic Parsing as Machine Translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 47–52. https://www.aclweb.org/anthology/P13-2009

[6] Yoav Artzi, Dipanjan Das, and Slav Petrov. 2014. Learning Compact Lexicons for CCG Semantic Parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1273–1283. https://doi.org/10.3115/v1/D14-1134

[7] Yoav Artzi, Nicholas Fitzgerald, and Luke Zettlemoyer. 2014. Semantic Parsing with Combinatory Categorial Grammars. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*. Association for Computational Linguistics, Doha, Qatar. https://www.aclweb.org/anthology/D14-2003

[8] Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping Semantic Parsers from Conversations. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., 421–432. https://www.aclweb.org/anthology/D11-1039

[9] Yoav Artzi and Luke Zettlemoyer. 2013. Weakly Supervised Learning of Semantic Parsers for Mapping Instructions to Actions. *Transactions of the Association for Computational Linguistics* 1 (2013), 49–62. https://doi.org/10.1162/tacl_a_00209

[10] Yoav Artzi and Luke S. Zettlemoyer. 2015. *Situated understanding and learning of natural language*. Ph.D. Dissertation.

[11] Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. 2017. DeepCoder: Learning to Write Programs. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[12] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for Sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, 178–186. https://www.aclweb.org/anthology/W13-2322

[13] Tal Ben-Nun, Alice Shoshana Jakobovits, and Torsten Hoefler. 2018. Neural Code Comprehension: A Learnable Representation of Code Semantics. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 3589–3601.

[14] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluating the Role of Bleu in Machine Translation Research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Trento, Italy. https://www.aclweb.org/anthology/E06-1032

[15] Donald D. Chamberlin and Raymond F. Boyce. 1974. SEQUEL: A Structured English Query Language. In *Proceedings of the 1974 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control* (Ann Arbor, Michigan) *(SIGFIDET '74)*. Association for Computing Machinery, New York, NY, USA, 249–264. https://doi.org/10.1145/800296.811515

[16] Shobhit Chaurasia and Raymond J. Mooney. 2017. Dialog for Language to Code. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, 175–180. https://www.aclweb.org/anthology/I17-2030

[17] Wanxiang Che, Yanqiu Shao, Ting Liu, and Yu Ding. 2016. SemEval-2016 Task 9: Chinese Semantic Dependency Parsing. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, 1074–1080. https://doi.org/10.18653/v1/S16-1167

[18] Xinyun Chen, Chang Liu, and Dawn Song. 2018. Tree-to-Tree Neural Networks for Program Translation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 2552–2562.

[19] James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving Semantic Parsing from the World's Response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, Uppsala, Sweden, 18–27. https://www.aclweb.org/anthology/W10-2903

[20] Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke Zettlemoyer, and Eduard Hovy. 2019. Iterative Search for Weakly Supervised Semantic Parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 2669–2680. https://doi.org/10.18653/v1/N19-1273

[21] Li Dong and Mirella Lapata. 2016. Language to Logical Form with Neural Attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 33–43. https://doi.org/10.18653/v1/

P16-1004

[22] Li Dong and Mirella Lapata. 2018. Coarse-to-Fine Decoding for Neural Semantic Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 731–742. https://doi.org/10.18653/v1/P18-1068

[23] Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence Modeling for Neural Semantic Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 743–753. https://doi.org/10.18653/v1/P18-1069

[24] Matt Gardner and J. Krishnamurthy. 2017. Open-Vocabulary Semantic Parsing with both Distributional Statistics and Formal Knowledge. In *AAAI*.

[25] Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence Driven Unsupervised Semantic Parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 1486–1495. https://www.aclweb.org/anthology/P11-1149

[26] Dan Goldwasser and Dan Roth. 2014. Learning from natural instructions. *Machine Learning* 94, 2 (1 Feb. 2014), 205–232. https://doi.org/10.1007/s10994-013-5407-y

[27] Justin Gottschlich, Armando Solar-Lezama, Nesime Tatbul, Michael Carbin, Martin Rinard, Regina Barzilay, Saman Amarasinghe, Joshua B Tenenbaum, and Tim Mattson. 2018. The Three Pillars of Machine Programming. arXiv:1803.07244 [cs.AI]

[28] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1631–1640. https://doi.org/10.18653/v1/P16-1154

[29] Sumit Gulwani and Prateek Jain. 2017. Programming by Examples: PL Meets ML. In *Programming Languages and Systems - 15th Asian Symposium, APLAS 2017, Suzhou, China, November 27-29, 2017, Proceedings (Lecture Notes in Computer Science, Vol. 10695)*, Bor-Yuh Evan Chang (Ed.). Springer, 3–20. https://doi.org/10.1007/978-3-319-71237-6_1

[30] Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2020. Neural Module Networks for Reasoning over Text. In *International Conference on Learning Representations*.

[31] Gary G. Hendrix, Earl D. Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. 1978. Developing a Natural Language Interface to Complex Data. *ACM Trans. Database Syst.* 3, 2 (June 1978), 105–147. https://doi.org/10.1145/320251.320253

[32] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021. Measuring Coding Challenge Competence With APPS. *arXiv preprint arXiv:2105.09938* (2021).

[33] Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics* 33, 3 (2007), 355–396. https://doi.org/10.1162/coli.2007.33.3.355

[34] Roshni Iyer, Yizhou Sun, Wei Wang, and Justin Gottschlich. 2020. Software Language Comprehension using a Program-Derived Semantics Graph. In *NeurIPS 2020 Workshop on Computer-Assisted Programming*. https://openreview.net/forum?id=AGLG_DgpE2l

[35] Srinivasan Iyer, Alvin Cheung, and Luke Zettlemoyer. 2019. Learning Programmatic Idioms for Scalable Semantic Parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 5425–5434. https://doi.org/10.18653/v1/D19-1545

[36] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 963–973. https://doi.org/10.18653/v1/P17-1089

[37] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2018. Mapping Language to Code in Programmatic Context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 1643–1652. https://doi.org/10.18653/v1/D18-1192

[38] Tim Johnson. 1984. Natural Language Computing: The Commercial Applications. *The Knowledge Engineering Review* 1, 3 (1984), 11–23. https://doi.org/10.1017/S0269888900000588

[39] Shoaib Kamil, Alvin Cheung, Shachar Itzhaky, and Armando Solar-Lezama. 2016. Verified Lifting of Stencil Computations. *SIGPLAN Not.* 51, 6 (June 2016), 711–726. https://doi.org/10.1145/2980983.2908117

[40] Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to Transform Natural to Formal Languages. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3* (Pittsburgh, Pennsylvania) *(AAAI'05)*. AAAI Press, 1062–1068.

[41] Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural Semantic Parsing with Type Constraints for Semi-Structured Tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 1516–1526. https://doi.org/10.18653/v1/D17-1160

[42] Jayant Krishnamurthy and Tom M. Mitchell. 2015. Learning a Compositional Semantics for Freebase with an Open Predicate Vocabulary. *Transactions of the Association for Computational Linguistics* 3 (2015), 257–270. https://doi.org/10.1162/tacl_a_00137

[43] S. Kulal, Panupong Pasupat, K. Chandra, Mina Lee, Oded Padon, A. Aiken, and Percy Liang. 2019. SPoC: Search-based Pseudocode to Code. In *NeurIPS*.

[44] Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling Semantic Parsers with On-the-Fly Ontology Matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA,

1545–1556. https://www.aclweb.org/anthology/D13-1161

[45] Royi Lachmy, Valentina Pyatkin, and Reut Tsarfaty. 2021. Draw Me a Flower: Grounding Formal Abstract Structures Stated in Informal Natural Language. *CoRR* abs/2106.14321 (2021). arXiv:2106.14321 https://arxiv.org/abs/2106.14321

[46] Mike Lewis and Mark Steedman. 2013. Combined Distributional and Logical Semantics. *Transactions of the Association for Computational Linguistics* 1 (2013), 179–192. https://doi.org/10.1162/tacl_a_00219

[47] Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning Dependency-Based Compositional Semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 590–599. https://www.aclweb.org/anthology/P11-1060

[48] Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent Predictor Networks for Code Generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 599–609. https://doi.org/10.18653/v1/P16-1057

[49] Peter C. Lockemann and Frederick B. Thompson. 1969. A Rapidly Extensible Language System (The REL Language Processor). In *International Conference on Computational Linguistics COLING 1969: Preprint No. 34*. Sånga Säby, Sweden. https://www.aclweb.org/anthology/C69-3401

[50] Sifei Luan, Di Yang, Celeste Barnaby, Koushik Sen, and Satish Chandra. 2019. Aroma: Code Recommendation via Structural Code Search. *Proc. ACM Program. Lang.* 3, OOPSLA, Article 152 (Oct. 2019), 28 pages. https://doi.org/10.1145/3360578

[51] Semantic Machines, Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-Oriented Dialogue as Dataflow Synthesis. *Transactions of the Association for Computational Linguistics* 8 (September 2020). https://www.microsoft.com/en-us/research/publication/task-oriented-dialogue-as-dataflow-synthesis/

[52] Zohar Manna and Richard Waldinger. 1980. A Deductive Approach to Program Synthesis. *ACM Trans. Program. Lang. Syst.* 2, 1 (Jan. 1980), 90–121. https://doi.org/10.1145/357084.357090

[53] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2020. Bao: Learning to Steer Query Optimizers. arXiv:2004.03814 [cs.DB]

[54] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A Learned Query Optimizer. *Proc. VLDB Endow.* 12, 11 (July 2019), 1705–1718. https://doi.org/10.14778/3342263.3342644

[55] Aditya Menon, Omer Tamuz, Sumit Gulwani, Butler Lampson, and Adam Kalai. 2013. A Machine Learning Framework for Programming by Example. In *Proceedings of the 30th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 28)*, Sanjoy Dasgupta and David McAllester (Eds.). PMLR, Atlanta, Georgia, USA, 187–195. http://proceedings.mlr.press/v28/menon13.html

[56] Maxwell Nye, Luke Hewitt, Joshua Tenenbaum, and Armando Solar-Lezama. 2019. Learning to Infer Program Sketches. , 4861–4870 pages. http://proceedings.mlr.press/v97/nye19a.html

[57] Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajič, Angelina Ivanova, and Yi Zhang. 2014. SemEval 2014 Task 8: Broad-Coverage Semantic Dependency Parsing. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics, Dublin, Ireland, 63–72. https://doi.org/10.3115/v1/S14-2008

[58] Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. 2020. Improving Compositional Generalization in Semantic Parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 2482–2495. https://doi.org/10.18653/v1/2020.findings-emnlp.225

[59] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[60] Panupong Pasupat and Percy Liang. 2016. Inferring Logical Forms From Denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 23–32. https://doi.org/10.18653/v1/P16-1003

[61] Hoifung Poon. 2013. Grounded Unsupervised Semantic Parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, 933–943. https://www.aclweb.org/anthology/P13-1092

[62] Hoifung Poon and Pedro Domingos. 2009. Unsupervised Semantic Parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1* (Singapore) *(EMNLP '09)*. Association for Computational Linguistics, USA, 1–10.

[63] Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to Code: Learning Semantic Parsers for If-This-Then-That Recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, 878–888. https://doi.org/10.3115/v1/P15-1085

[64] Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract Syntax Networks for Code Generation and Semantic Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 1139–1149. https://doi.org/10.18653/v1/P17-1105

[65] Jonathan Ragan-Kelley, Connelly Barnes, Andrew Adams, Sylvain Paris, Frédo Durand, and Saman Amarasinghe. 2013. Halide: A Language and Compiler for Optimizing Parallelism, Locality, and Recomputation in Image Processing Pipelines. *SIGPLAN Not.* 48, 6 (June 2013), 519–530. https://doi.org/10.1145/2499370.2462176

[66] Alexander J. Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and C. Ré. 2019. Snorkel: rapid training data creation with weak supervision. *The Vldb Journal* 29 (2019), 709 – 730.

[67] Veselin Raychev, Pavol Bielik, and Martin Vechev. 2016. Probabilistic Model for Code with Decision Trees. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications* (Amsterdam, Netherlands) *(OOPSLA 2016)*. Association for Computing Machinery, New York, NY, USA, 731–747. https://doi.org/10.1145/2983990.2984041

[68] Veselin Raychev, Martin Vechev, and Andreas Krause. 2015. Predicting Program Properties from "Big Code". In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (Mumbai, India) *(POPL '15)*. Association for Computing Machinery, New York, NY, USA, 111–124. https://doi.org/10.1145/2676726.2677009

[69] Eui Chul Richard Shin, Miltiadis Allamanis, Marc Brockschmidt, and Alex Polozov. 2019. Program Synthesis and Semantic Parsing with Learned Code Idioms. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 10824–10834. https://proceedings.neurips.cc/paper/2019/hash/cff34ad343b069ea6920464ad17d4bcf-Abstract.html

[70] Armando Solar-Lezama. 2008. *Program Synthesis by Sketching*. Ph.D. Dissertation. USA. Advisor(s) Bodik, Rastislav.

[71] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. 2006. Combinatorial Sketching for Finite Programs. *SIGARCH Comput. Archit. News* 34, 5 (Oct. 2006), 404–415. https://doi.org/10.1145/1168919.1168907

[72] Mark Steedman. 1987. Combinatory grammars and parasitic gaps. *Natural Language & Linguistic Theory* 5 (1987), 403–439.

[73] Mark Steedman and Jason Baldridge. 2007. Combinatory Categorical Grammar.

[74] Sanjay Subramanian, Ben Bogin, Nitish Gupta, Tomer Wolfson, Sameer Singh, Jonathan Berant, and Matt Gardner. 2020. Obtaining Faithful Interpretations from Compositional Neural Networks. In *ACL*.

[75] Ngoc Tran, Hieu Tran, Son Nguyen, Hoan Nguyen, and Tien Nguyen. 2019. Does BLEU Score Work for Code Migration? *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)* (May 2019). https://doi.org/10.1109/icpc.2019.00034

[76] David L. Waltz. 1978. An English Language Question Answering System for a Large Relational Database. *Commun. ACM* 21, 7 (July 1978), 526–539. https://doi.org/10.1145/359545.359550

[77] Catherine Wong, Kevin Ellis, Joshua B. Tenenbaum, and Jacob Andreas. 2021. Leveraging Language to Learn Program Abstractions and Search Heuristics. *Proceedings of the 38th International Conference on Machine Learning* (2021). https://arxiv.org/abs/2106.11053

[78] Yuk Wah Wong and Raymond Mooney. 2006. Learning for Semantic Parsing with Statistical Machine Translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*. Association for Computational Linguistics, New York City, USA, 439–446. https://www.aclweb.org/anthology/N06-1056

[79] W. A. Woods. 1973. Progress in Natural Language Understanding: An Application to Lunar Geology. In *Proceedings of the June 4-8, 1973, National Computer Conference and Exposition* (New York, New York) *(AFIPS '73)*. Association for Computing Machinery, New York, NY, USA, 441–450. https://doi.org/10.1145/1499586.1499695

[80] Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based Structured Prediction for Semantic Parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1341–1350. https://doi.org/10.18653/v1/P16-1127

[81] Fangke Ye, Shengtian Zhou, Anand Venkat, Ryan Marcus, Nesime Tatbul, Jesmin Jahan Tithi, Niranjan Hasabnis, Paul Petersen, Timothy Mattson, Tim Kraska, Pradeep Dubey, Vivek Sarkar, and Justin Gottschlich. 2020. MISIM: A Novel Code Similarity System. arXiv:2006.05265 [cs.LG]

[82] Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic Parsing for Single-Relation Question Answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, 643–648. https://doi.org/10.3115/v1/P14-2105

[83] Pengcheng Yin and Graham Neubig. 2017. A Syntactic Neural Model for General-Purpose Code Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Vancouver, Canada, 440–450. https://doi.org/10.18653/v1/P17-1041

[84] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 3911–3921. https://doi.org/10.18653/v1/D18-1425

[85] John M. Zelle and Raymond J. Mooney. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2* (Portland, Oregon) *(AAAI'96)*. AAAI Press, 1050–1055.

[86] Luke Zettlemoyer and Michael Collins. 2007. Online Learning of Relaxed CCG Grammars for Parsing to Logical Form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Association for Computational Linguistics, Prague, Czech Republic, 678–687. https://www.aclweb.org/anthology/D07-1071

[87] Luke S. Zettlemoyer and Michael Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence* (Edinburgh, Scotland) *(UAI'05)*. AUAI Press, Arlington,

Virginia, USA, 658–666.

[88] Yunming Zhang, Mengjiao Yang, Riyadh Baghdadi, Shoaib Kamil, Julian Shun, and Saman Amarasinghe. 2018. GraphIt: A High-Performance Graph DSL. *Proc. ACM Program. Lang.* 2, OOPSLA, Article 121 (Oct. 2018), 30 pages. https://doi.org/10.1145/3276491

A NATURAL LANGUAGE SEMANTIC PARSING

Natural language (NL) analysis can be segmented into at least two fundamental categories: *syntax* and *semantics*. In general, *syntax* is the way a natural language phrase is structured: the order and arrangement of words and punctuation. The *semantics* is the meaning that can be derived from such syntax. For example, the two sentences *"the boy cannot go"* and *"it is not possible for the boy to go"*, are semantically equivalent even though they are syntactically different. By lifting semantics from syntax, NL semantic parsers can map semantically equivalent sentences to ideally the same logical form, even when they have different syntaxes. An example of this is shown in Figure 1b, where two syntactically different sentences are shown to be semantically equivalent using an abstract meaning representation (AMR) as its logical form (more details in Section A.3). Note that most existing meaning representations are not perfect, in that they may not always map semantically-equivalent but syntactically-divergent inputs to the same logical form.

Historically, NL semantic parsers have enabled computers to query databases [84], play card games [26], and even act as conversational agents such as Apple's Siri and Amazon's Alexa. In this section, we discuss how NL semantic parsers extract meaning from natural language *utterances* into various machine-understandable representations.

A.1 Components of a Semantic Parsing System

Given a NL input, a NL semantic parsing system generates its semantic representation as a structured logical form. These logical forms are also known as *meaning representations* or *programs*. To generate these outputs, semantic parsing pipelines are usually trained by some learning algorithm to produce a distribution over the output search space, then search for a best scoring logical form in that distribution. This logical form can then be executed against some environment to carry out an intended task (e.g. to query a database). Because these logical forms are generally designed to be machine-understandable representations, they tend to have a structured nature, following an underlying formalism by which some grammar can be used to derive valid logical forms.

A.2 Grammars

Grammar in the context of natural language processing is a set of rules that governs the formation of some structure (the parsing of NL utterances) to ensure well-formedness. A set of grammatical rules can constrain the search space of possible outputs. Combinatory categorical grammar (CCG) [72, 73] is one such example of a popular grammar formalism. It has historically made frequent appearances in semantic parsing models [44, 86, 87] due to its ability to jointly capture syntactic and semantic information of the constituents which it label. Consider the example in Figure 1c. By pairing syntactic and semantic information, CCGs can describe textual input for a machine to understand the semantics with the syntax. In addition, the prominence of CCG in natural language processing efforts allows a model using CCG to leverage the semantic understanding within that formalism, and the large body of existing CCG work.

A.3 Meaning Representations

Meaning representations are the end product of NL semantic parsing. The output logical formula, or meaning representation, grounds the semantics of the natural language input in the target machine-understandable domain. A simple example of a logical form is a database query. Popular datasets explored in semantic parsing include Geo880, Jobs640,

```
I'd like to book a flight from San Diego to Toronto
```

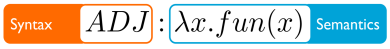
```
SELECT DISTINCT f1.flight_id
FROM flight f1, airport_service a1, city c1,
  airport_service a2, city c2
WHERE f1.from_airport = a1.airport_code
  AND a1.city_code = c1.city_code
  AND c1.city_name = 'san diego'
  AND f1.to_airport = a2.airport_code
  AND a2.city_code = c2.city_code
  AND c2.city_name = 'toronto';
```

(a) Sample SQL query for a flight reservation (credit: Iyer et al. [36]).

```
(p / possible
   :domain (g / go-01
              :arg0 (b / boy))
   :polarity -))

The boy cannot go.
It is not possible for the boy to go.
```

(b) Abstract meaning representation of two semantically equivalent but syntactically different sentences (credit: Banarescu et al. [12]).



(c) An example of a CCG category. Syntax (the ADJ represents an adjective part of speech) is paired with semantics (the lambda term expression) (credit: Artzi et al. [7]).

Fig. 1. Meaning representation and grammar examples for semantic parsing.

ATIS, and SQL itself [20, 36, 84]. An example of a SQL query is shown in Figure 1a. Other classes of semantic parsing outputs include NL instructions to actions in an environment and lambda calculus expressions. In most of these cases, the semantic parser is specific to a particular domain and may not generalize to out-of-domain inquiries or differing data structures.

Meaning representations can also take on graph-based forms. Popular graphical representations are the Abstract Meaning Representation [12], Semantic Dependency Parsing [17, 57], Universal Conceptual Cognitive Annotation [1], Dependency-based Compositional Semantics [47], and CCGBank [33].

While most general purpose programming languages are not necessarily abstract in their representations of intention (source code often involves details about variables, memory allocation, class structure, etc.), code can serve as a machine-understandable logical forms. To lift semantics from the code's syntax, recent works have gone into developing and using meaning representations of code, such as the abstract syntax tree (AST) [2–4, 63, 67, 68], sketching [70], simplified parse tree (SPT) [50], contextual flow graph (XFG) [13], context-aware semantic structure (CASS) [81], and program-derived semantics graph (PSG) [34]. These representation structures share the common goal of being a vehicle of machine-understandable meaning of the code, abstracted to some extent from the syntax of code. We discuss the application of some of these structures in Section B.

B PROGRAM SYNTHESIS / CODE GENERATION

In this section, we review a subset of existing efforts for program synthesis and code generation. Program synthesis is the task of generating software from some high-level program specification [52]. The purpose of this section is to highlight parallels between this process of using a high-level program specification to generate code, and NL semantic parsing's translation of natural language into meaning representations.

Code has many different dimensions. Despite what could be construed as a strict syntax for compile-ability, source code can be reasoned about on many different levels: semantics, runtime, memory footprint, and compiler details, to name a few. As a result, design and selection of a code representation structure, and the subsequent reasoning about said structure is an interesting and nuanced task. This challenge is similar to the consideration of meaning representation in semantic parsing.

Many code recommendation systems use code similarity to retrieve different implementations of the same intention. Semantic code representations can provide a foundation on which to extract code similarity [34, 50, 81]. Program

synthesis has also come in the flavor of machine translation [18, 63, 78], pseudocode-to-code transformation [43], and inductive program synthesis by input-output examples [11, 29, 55].

The task of program synthesis tends to be more tractable if the output programming language is less complex, making the output space more compact. Declarative programming languages such as SQL [15], Halide [65], and GraphIT [88] express the intention of a computation without detailing its execution. For this reason, these types of languages are also called *intentional programming languages* in the field of machine programming. As a function of the syntax and semantics being closely related, programs written in intentional programming languages tend to be easier to reason about in a semantic space. A byproduct of this phenomenon is that intentional programming languages can also be easier to optimize. Neo [54] and Bao [53] are learned neural models that optimize SQL queries. Verified lifting [39] also demonstrates an ability to optimize Halide code.

C EVOLUTION

C.1 Grammar in Semantic Parsing

Supervised learning approaches have induced grammars to represent the underlying semantics of a knowledge base. Zelle and Mooney [85] present a system that employs inductive logic programming methods to *learn* shift-reduce parsers that map sentences into database queries. Kate et al. [40] also induce and recursively apply transformation rules to map natural language inputs (or their syntactic parse trees) to a parse tree in the target representation language.

In the literature, CCGs have been a popular and powerful tool to capture textual semantics by mapping sentences to their logical forms [6, 8, 9, 86, 87]. Zettlemoyer and Collins [87] present a supervised on-line learned CCG model to parse sentences into lambda-calculus representations of their underlying semantics [86]. The proposed learning algorithm induces a grammar (CCG) and log-linear parsing model that represents a distribution over syntactic and semantic analyses of a given sentence under that CCG. However, a weakness of this model is the inability of the learner to parse complex sentences for which their lexical generator does not create lexicon entries. Artzi and Zettlemoyer [10] introduce an approach to induce a grammar then use corpus-level statistics at each lexicon update during learning to prune the induced CCG lexicons [7]. Performance improvements in this system indicate an advantage to maintaining compact CCG lexicons. The positive impact of compact grammars on NL semantic parsers hints that compact grammars may also help code generation systems.

A challenge in semantic parsing has been how to scale a semantic parsing model for different domains. Kwiatkowski et al. [44] present a learned ontology matching model that adapts the output logical form of the semantic parser for each target domain. The parser uses a probabilistic CCG to construct a domain-independent meaning representation from the NL input, then uses the ontology matching model to transform that representation into the target domain. This underspecified intermediate logical form allows the model to share grammar structure across domains instead of re-learning different grammars for each target ontology.

D SUPERVISION IN SEMANTIC PARSING

D.1 Supervision by Denotation

An alternative to fully-supervised training for semantic parsing systems is supervision by denotation. Clarke et al. [19] present a learning algorithm that relies only on a binary feedback signal for validation, and Liang et al. [47] also elide the need for labeled logical forms by inducing a NL semantic parsing model from question-answer pairs. The resulting

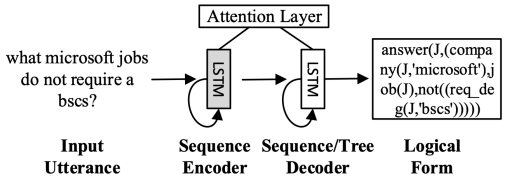


Fig. 2. Encoder-decoder model for mapping NL to logical forms (Credit: Dong and Lapata [21]).

parsers are competitive with fully supervised systems in the same domain, demonstrating the feasibility of supervision by denotation.

However, two primary challenges that arise with supervision by denotation are *(1)* computational intractability in the intermediate latent representations and *(2)* increased potential for spurious logical forms. Modifications to supervision by denotation have been proposed to address these challenges. Goldwasser and Roth [26] identify the need to constrain the intermediate representation space. By capturing alignments between NL instruction textual fragments and meaning representation logical fragments, the model can more closely interpret NL instructions and hopefully reduce the potential for spurious forms. Dasigi et al. [20] propose an iterative training algorithm that internally manages the problem of spuriousness by alternating between searching for logical forms and maximizing the marginal likelihood of retrieved logical forms. This semi-supervised training scheme provides guidance to progressively search for logical forms of increasing complexity, making it more likely the model is "correctly" learning. Pasupat and Liang [60] propose a system that performs dynamic programming on denotations to address the issue of exploding latent search space and spurious results. By observing that the space of possible denotations grows more slowly than the space of logical forms, this dynamic programming approach can find all consistent logical forms up to some bounded size. Then, because spurious logical forms will generally give a wrong answer once the context changes, they generate a slightly modified context and examine whether the logical form produces the correct denotation in this alternate context. If it does not, this spurious logical form can be discarded. The combination of limited search space and pruning of spurious forms allows them to use denotation as a stronger supervision signal. These techniques in systems for inductive program synthesis from input-output examples may be adaptable to the code generation and machine programming domains.

D.2 Weak Supervision and Self Supervision

Weakly-supervised learning [1] and self-supervised learning are presented as more scalable approaches to semantic parsing [25]. Artzi and Zettlemoyer [8] induce a NL semantic parser by using conversational feedback from un-annotated conversation logs to model the meaning of user statements with latent variables. They then measure how well a candidate meaning representation matches their expectations about the conversation context and dialog domain. Poon and Domingos [62] present a self-supervised system for NL semantic parsing by recursively clustering semantically equivalent logical forms to abstract away syntactical implementation details. This method, however, does not ground the induced clusters in an ontology. Poon [61] subsequently proposes a *grounded* unsupervised learning approach that takes a set of natural language questions and a database, learns a probabilistic semantic grammar over it, then constrains the search space using the database schema. Likewise, recent works in open vocabulary NL semantic parsing replace a formal knowledge base with a probabilistic base learned from a text corpus [24, 42, 46]. In this framework, the predicates (parts of the structure that make claims about the actions or characteristics of the subject) of the grammar are derived

[1] We discuss other emerging works of weak supervision, such as Snorkel [66], in Section 5.

directly from the domain text. The model must also learn the execution models for these induced predicates [42, 46]. To address the challenge of generalizing NL semantic parsing models to out-of-domain and open-vocabulary applications, Gardner and Krishnamurthy [24] map language to a weighted combination of queries rather than a single knowledge base query, in order to represent a much broader class of concepts. This approach can expand the applicability of semantic parsing techniques to more complex domains.

E MODERN ADVANCES IN NEURAL SEMANTIC PARSING FOR CODE GENERATION

In this section, we discuss modern advances in neural models for semantic parsing of natural language utterances to code generation.

E.1 Context and Variable Copying

Iyer et al. [37] develop an architecture that leverages programmatic context to write contextually-relevant code. This encoder-decoder system feeds the encoder the input utterance along with featurized representations of environment identifiers, such as type and name of each variable and method. The decoding process attends first to the NL input, then matches words in the NL text to the environment identifier representations. Additionally, a supervised copy mechanism [28] based on attention weights is presented to copy an environment token to the output. This allows the model to copy in variable names not seen in the training data or in the input utterance.

E.2 Grammar in Neural Semantic Parsing Models

Grammar can guide and constrain the decoding process of neural encoder-decoder models for quicker training and more accurate results. Xiao et al. [80] propose a sequence-based approach to generate a knowledge base query: they sequentialize the target logical forms, then demonstrate the advantage of incorporating grammatical constraints into the derivation sequence. Likewise, when querying information from semi-structured tables, Krishnamurthy et al. [41] demonstrate that enforcing type constraints and including explicit entity linking can bolster accuracy because it ensures the generation of well-typed logical forms.

As discussed in Section B, code is dynamic in its representation and semantics, making grammar for generating code especially interesting. Unlike prior works which focus on domain-specific languages, which may have limited scope and complexity, Yin and Neubig [83] and Ling et al. [48] present methods for generating high-level general-purpose programming language (GPL) code such as Python. Ling et al. [48] present a data-driven sequence-to-sequence code generation method that implements multiple predictors: a generative model for code generation, and multiple pointer models to copy keywords from the input. Building off of Ling et al. [48], Yin and Neubig [83] point out that GPLs are generally more complex in schema and syntax than other semantic parsing domains. Therefore, code generation models could benefit from a grammar that explicitly captures the target programming language syntax as prior knowledge. Yin and Neubig [83] do this by designing their model to only generate a series of actions that can produce a valid abstract syntax tree (AST). The actions themselves are derived from the underlying syntax of the programming language. This approach demonstrates improved Python code generation, but Yin and Neubig point out a decline in performance as the AST becomes more complex, noting that there may be space for improvement in scalability.

As highlighted in Section C.1, compact grammar lexicons have been shown to improve semantic parsing performance [7]. Code idioms are bits of code that occur frequently across a code repository and tend to have a specific semantic purpose. They can be incorporated into grammars to reduce the burden on training data for semantic parsing of grammar-constrained systems [35]. Frequently-seen idioms can be used to collapse the grammar for decoding, making

a compact set of rules that supervise the learning of the semantic parser. Another advantage to mining code idioms is that they may encourage simultaneous high-level and low-level reasoning in the model [69]. Mined code idioms are incorporated into the grammar as new named operators, allowing the model to emit entire idioms at once, thus generating high-level and low-level program constructs interchangeably at each step.

E.3 Sketching

Dong and Lapata [22] present a structure-aware semantic parsing decoder that employs sketching to generate meaning representations in two phases: *(1)* generating a coarse sketch of the meaning representation that abstracts away low-level details, and *(2)* filling in those missing details of arguments and variable names. An attention-enhanced RNN serves as the fine meaning decoder, using the coarse sketch to constrain the decoding output. This approach additionally does not need syntax or grammatical information for the output space because it is naturally constrained by the sketch.

Nye et al. [56] present a method to dynamically integrate natural language instructions and examples to infer program sketches. Previous systems have mostly used static, hand-designed intermediate sketch grammars, which tend to be rigid in how much a system relies on pattern recognition vs. symbolic search. When given an "easy" or familiar task, Nye et al. [56]'s neuro-symbolic synthesis system can rely on learned pattern recognition to produce a more complete output. When given a "hard" task, the system can produce a less complete sketch and spend more time filling in that sketch using search techniques. By learning how complete to make the sketch for different applications, this model can employ sketching in a more fitting and customized manner.