From Ontologies to Learning-Based Programs

Quan Guo[1] , **Andrzej Uszok**[2] , **Parisa Kordjamshidi**[2,3]
[1]Tulane University
[2]Florida Institute for Human and Machine Cognition
[3]Michigan State University
qguo2@tulane.edu, auszok@ihmc.us, kordjams@msu.edu

Abstract

In this paper, we discuss the work in progress on designing a prototype for a novel declarative learning-based programming system and present our preliminary results. The main idea is to express learning-based programs in terms of declarative domain knowledge. Given that the existing ontologies contain rich domain and world knowledge, we propose to automatically generate the learning-based programs from the current ontology representation languages such as OWL. The ontological concepts and domain relationships are compiled to a graph which is a partial program. The nodes in the graph are connected to data sensors and learners. Local training algorithms can use data and train learners corresponding to each concept and domain relationship in the graph. Global inference mechanisms make the final decisions based on the local prediction of the learners and under the ontological constraints. We test our framework on the entity-mention-relation extraction task.

1 Introduction

Nowadays, experts in various problem domains try to use machine learning and data-driven solutions for real-world problems. There are many tools and languages that help to design intelligent systems that use learning as their main component. The goal of these tools is to provide a high level of abstraction for designing learning models that make the low-level algorithmic details transparent from the users perspective. The most ambitious abstractions are the ones that are in terms of logical representations of domain concepts and relationships and hide all the computational units from the users [Kordjamshidi *et al.*, 2018]. This is different from most commonly used frameworks such as current deep learning tools such as pyTorch[1] and TensorFlow[2]. In the current tools, users have access to high-level functions to design deep learning architectures in terms of linear and nonlinear layers from which automatic derivations are taken to optimize the parameters. However, the connection between the application and the designed architectures is not a part of the program, and it is only known by the programmer who knows both the application and the design language for machine learning. Declarative learning-based programming [Kordjamshidi *et al.*, 2018; Kordjamshidi *et al.*, 2015] argues for abstractions that are based on the conceptual problem specification rather than the underlying computational units. On a related issue, most of the existing tools do not support expressing and using the structure of the data directly. There is no principled way for expressing the complex relational data and connecting it to the learning architectures. Dealing with the structure and figuring out how to use it in learning is left to the user.

Though the recent graph networks [Zhou *et al.*, 2018] are capable of representing graph structures, their graphs specify complicated data instances rather than a high -level specification of the problem domain. There is no way to express and use first-order knowledge and the relational schema of the data explicitly. There is a gap between the abstraction level that is used in applications and machine learning models. The question that we address here is *How to bridge the gap between domain knowledge and learning-based programs?*

The main idea of this paper is to use domain ontologies as representative of the application domain in the learning-based programs. Ontologies are a rich source of the domain and world knowledge. They are powerful in representing concepts and their relationships for an application. They are a means to express first-order knowledge about groups of object and types of relationships which is impossible to express in data instances. Here, we assume the ontologies contain the domain information that specifies the application's central concepts, relationships and the constraints over them. Therefore, we generate learning-based programs by automatic compilation of the ontology of the domain. We bridge between the two levels of abstractions in the application domain and machine learning in a modular way. We address the problem of bridging the ontologies to learning-based programming by proposing an end-to-end pipeline with the following components,

1. A schema representation of the domain concepts and their relationships is generated automatically from the standard ontology languages (here web ontology lan-

[1]https://pytorch.org
[2]https://www.tensorflow.org

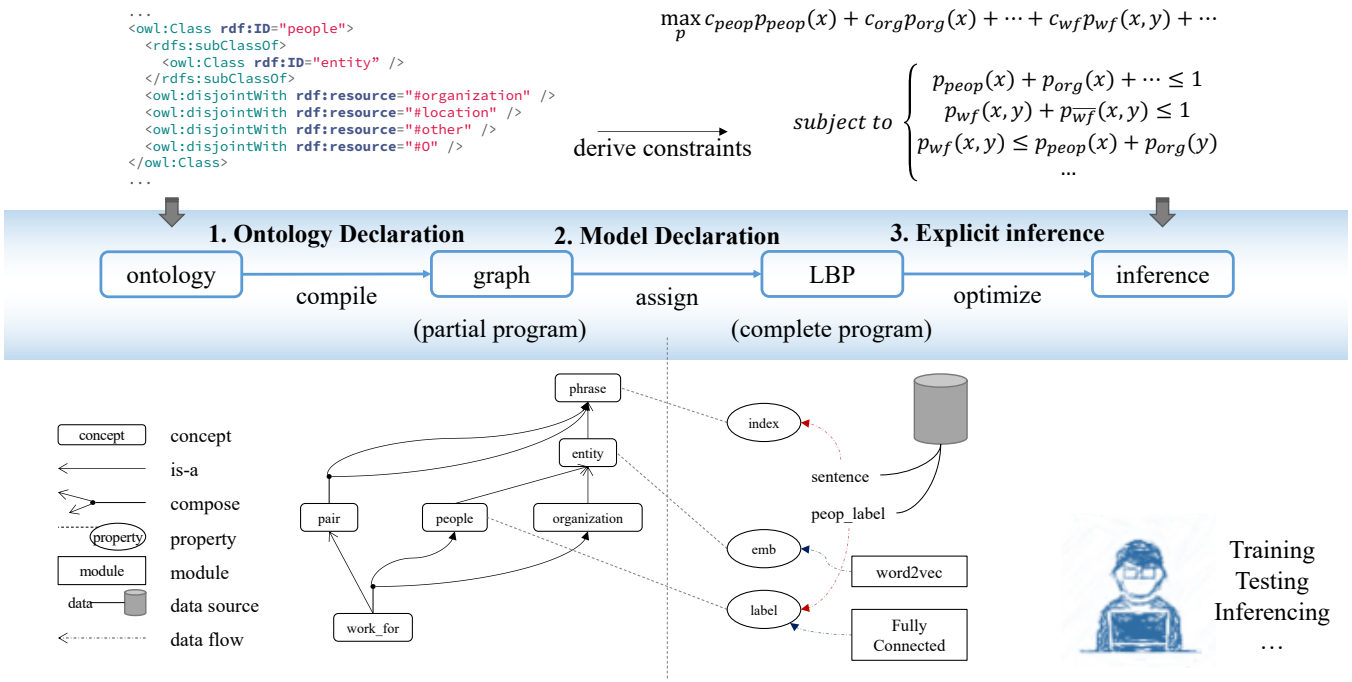

Figure 1: We propose a pipeline from ontology to learning-based program. It consists of three explicit steps: 1. Ontology Declaration, 2. Model Declaration, and 3. Explicit Inference.

guage OWL[3]) and form the backbone graph of the target learning-based program.

2. The generated graph concepts are associated with data and learning models (we call them concept learners),
3. The learning targets and loss functions are derived automatically.
4. The final predictions are made based on the concept learners and global ontological constraints.

To showcase the effectiveness of the current framework and the components of our pipeline, we use the entity-mention-relation extraction (EMR) task and validate on CoNLL data[4]. The task is as follows: **given** an input text such as "*Washington works for Associated Press.*", **find** a model that is able to extract the semantic entity types (e.g., people, organizations, and locations) as well as relations between them (e.g., works for, lives in), and **generate** the following labels for entities: *[Washington]*$_{people}$ *works for [Associated Press]*$_{organization}$, and find out the relationship between *Washington* and *Associated Press* is *worksFor*.

2 Main Components

The proposed pipeline includes three major components: the ontology declaration, the model declaration, and explicit inference. Figure 1 shows this pipeline. Ontology declaration, shown as step 1 in the figure, provides the domain concepts and the knowledge about the relationships between these concepts. We take a domain ontology in the standard ontology language OWL as the input to our pipeline. The top-left snippet of code is the OWL declaration of "people", considered a subclass of "entity", and is disjoint with organization, location, other (entities), and non-entities. The ontology is compiled automatically and generates the partial body of the learning based program as a graph. Some parts of the graph is shown on the bottom-left of Figure 1. The graph is consist of concepts, connections, and properties. Connections indicate the way concepts relate to each other. Properties are named values attached to concepts, which will later be provided by sensors and learners. The partial program can be completed by the programmer with the model declaration. Model declaration is to assign sensors and learners to properties of ontological concepts to form a complete program. Sensors are snippets of the program to read from the dataset, while learners are parameterized computation modules. Explicit inference optimizes the prediction taking into account the constraints extracted from the ontology as examples shown on top-right of Figure 1. Explicit inference makes the final prediction consistent with the domain knowledge.

With the complete learning-based program compiled, assigned, and optimized, the user can interactively train the parameters, test the performance, or just do inference over specific samples. In fact, the ontology plays the role of the skeleton of the learning-based program in many ways. Firstly, it

[3]https://www.w3.org/TR/owl-overview/
[4]https://www.clips.uantwerpen.be/conll2003/ner/

declares the domain itself and forms the initial body of the program. Secondly, it allows the learning models to be built upon its properties, making the source and destination of each learner clear and connect those tightly to the ontological concepts. Thirdly, it serves as a source of global knowledge based on which we can make joint inference for prediction of concepts and relationships. These three components are detailed below.

2.1 Ontology declaration

The structure of the domain knowledge is represented via ontologies including concepts, relationships, and properties. In this work we assume the ontology of the domain is given in OWL language which is among the most popular languages to express ontologies. We compile a given ontology from OWL format into a directed graph $\mathcal{G}(V, E)$, where the nodes are denoted by $V = \{v_1, v_2, \cdots, v_n\}$ and the edges by $E = \{e = (v_i, v_j) | v_i, v_j \in V\}$.

All domain concepts and relationships are represented in nodes and the edges are related to a predefined set of semantics about the way the concepts are connected to each other. This includes "is-a", "not-a", and "has-a" edges. The "is-a" connection represents inheritance between two concepts, which is the most common connection in ontology modeling. The implied semantic is that the validity of the source concept indicates the validity of the destination concept. The "has-a" connection represents that the source concept is composed of the destination concept(s). The semantic of "has-a" is that the validity of the source concept indicates the validity of all destination concepts at the same time. For example, in "people 'work for' organization", where "work for" is a concept composed of "people" and "organization". We call the concepts like "work for" the composed concepts in our graph. They can be defined by "has-a" connection to the components. As shown in Figure 1, the composed concept "work for" is connected to the first argument "people" and the second one "organization", with two "has-a" connections.

Each concept is described with a collection of named properties $P_i = \{\text{name}_1 : p_1, \text{name}_2 : p_2, \cdots\}$.

```
from regr import Concept
# ...
people = Concept(name='people')
people.is_a(entity)
people.not_a(organization)
# ...
work_for = Concept(name='work_for')
work_for.is_a(pair)
work_for.has_a(people, organization)
```

2.2 Model declaration

Model declarations are made manually by adding sensors and learners to the partial learning-based program. It should be noticed that the partial learning-based program derived from the ontology, compiled from OWL or written as the graph, does not contain any sensor or learner to interact with data or computations. Sensors are program components that interact with the data source. Sensors read plain text, structured data, or features extracted by external pre-processing modules. They provide access to the data as well as the ground-truth labels. Learners are computational components that produce representations of known properties or predict the unknown ones. Learners usually come with parameters, and it is where the learning of the program will take place. The purpose of model declarations is threefold: connection to the input, connection to the computational units, and specifying the output. Firstly, it defines how data will be populated into the graph by connecting sensors to the properties of concepts (See the below Python snippet).

```
from regr.sensor import *
sentence['raw'] = ReaderSensor('sentence')
people['label'] = ReaderSensor('Peop')
```

We connect the property of an individual concept to a data sensor. At run-time, when the specific property is queried, the program collects data, usually in batches, from the corresponding sensor. It can also be the ground truth labels (*i.e.* , target values) that are used to train the learners.

Secondly, it defines how the computational modules will be connected to the concept properties. Learners with parameters, like neural network modules, are assigned to properties of concepts. The learners play the role of unknown transformations between properties. At run-time, when the specific output property is queried, the graph invokes the module to make a prediction.

```
from regr.sensor import *
phrase['w2v'] = EmbedderLearner('phrase',
  ↪  embedding_dim, sentence['raw'])
entity['emb'] = RNNLearner(embedding_dim,
  ↪  phrase['w2v'])
people['label'] = LogisticRegressionLearner(
  ↪  embedding_dim * 2, entity['emb']))
```

The above snippet assigned a learner of "EmbedderLearner" followed by a bidirectional gated recurrent unit (GRU) "RNNLearner" to the "emb" property of concept "entity". The "emb" property will be further used by a classifier "LogisticRegressionLearner" with log scale output, which is further assigned to "label" property of "people". Prediction will be made by the classifier based on its parameters. This learner can update its parameters in training process.

Lastly and most importantly, the learning target, *a.k.a.* the loss function of the learning-based program can be derived from the model declaration directly. We extract the set of the properties that are assigned to multiple sensors or learners at the same time, namely the properties of interest (POI). The loss function is defined automatically by checking the consistency of different sensor/learner assignments of properties in POI. One can assign a ground-truth sensor to a concept's label property, and then another prediction learner to the same property. The loss function is derived from the difference of the output of ground-truth sensor and the prediction learner. The parameters of the learners are trained by optimizing the derived loss with gradient descent algorithms. In other words, the assigned sensors provide data and ground truth labels to compute the loss, and the error will be back propagated to all the learners including classifiers that make prediction directly, or non-linear transformations that produce intermediate representations.

2.3 Explicit inference

Though the modules are trained to produce consistent predictions, this consistency cannot be guaranteed by the learners themselves. Learners may share input features but predict labels independently. The global consistency of the outputs is not guaranteed when the structural information over those is not used explicitly. To enforce the consistency of predictions, and more importantly to exploit the knowledge expressed in the input ontology, we introduce the explicit inference based on the ontological constraints.

We convert the ontological constraints over the concept sensors and learners into an integer linear programming (ILP) problem and maximize the overall confidence of the truth values of all concepts while satisfying the global constraints [Chang *et al.*, 2012; Roth and Yih, 2007]. We derive constraints from the input ontology automatically. In our EMR example, two types of constraints are considered: the disjoint constraints and the composed-of constraints. The disjoint constraints, interpreted from "not-a" connections between concepts, consider the mutual exclusivity of assigning concept labels to nodes. For example, an "entity" could not be a "people" and an "organization" at the same time. In EMR problem we assume at maximum one type of relationship can hold between two entities. The composed-of constraints, interpreted from "has-a" connections, constrain domain relationships and their arguments, stating that if a relationship, for example "work for", holds between two entities then the type of the arguments should satisfy that the first entity is "people" and the second is "organization". By solving the generated ILP problem, we can obtain a set of predictions that considers the structure of the data and the knowledge that is expressed in the domain ontology.

Experiments. In EMR task, with our handcrafted simple OWL ontology, we combined simple learners with raw features. We employ 50 dimensional word2vec feature [Pennington *et al.*, 2014], POS-tags, and dependency parsing roles[5] for words representations. We use a 5-gram and bi-directional gated recurrent units layer are used for sequence encoding. Each individual concept are predicted based on the encoding with a logistic regression. For pairs, we first create a 48-dimensional compact representation based on the encoding. We concatenate the representation of two words in a pair for the representation of the pair. We also encoded the distance between two words in the pair in log scale. When the word for the first argument comes after that for the second, we assign a negative distance. We further added the dependency relationship and lowest common ancestor as the pair features. Composed concepts are predicted based on pair features by local logistic regression learners. Inference is done globally on the top of all logistic regression prediction results of both single and composed concepts.

Table 1 shows the results of EMR example. We did not fit our model parameters specifically and used only a few linguistic features. Our goal is to demonstrate our language for designing models, features, and global inference, rather than to compete with the state-of-the-art of EMR task.

[5]Pos-tags and dependency parsing are extracted by spaCy: https://spacy.io/.

Table 1: EMR experimental results.

	(-)	(+)
location	0.8583	**0.8622**
organization	**0.7541**	0.7537
other	0.6639	**0.6680**
people	0.9145	**0.9165**
kill	0.5383	**0.5418**
live-in	0.4582	**0.4670**
located-in	0.4793	**0.4816**
orgbase-on	**0.4499**	0.4449
work-for	**0.4420**	0.4260

(-) Learner prediction without inference
(+) Global output with inference

We see significant evidence that global inference improves the results of independent learners, which is consistent with the previous results on this task [Kordjamshidi *et al.*, 2015; Zhang *et al.*, 2016]. Most of the results are improved after inference. However, "organization" and its related composed concepts, namely "orgbase-on" and "work-for", are negatively affected after inference.

Global inference can be helpful in multiple ways. Firstly, global inference eliminates the false positives and improves precision by disjoint constraints, which is indicated by "not-a" connections in the graph. Second, though in our current setting the baseline prediction is weak, it still gives a more consistent global result by resolving conflicts in the predictions. Thirdly, the inference balances between weak and strong learners in the model. It tends to give more improvement in weak learners than degeneration on strong ones. In practice, the weak performance of the learners could be due to the difficult nature of the concepts. It means it is difficult to improve their performance from a single learner. For example, learning the composed concept (*e.g.* "work for") is more difficult than those of simple concepts (*e.g.* "people"). Global inference balances between these learners. It improves the results of difficult concepts (from weak learners).

3 Related Works

The idea of learning-based programming is proposed in [Roth, 2005] for the first time and followed up in [Rizzolo and Roth, 2010; Rizzolo, 2011] by developing Learning-based Java programming paradigm which was designed to support machine learning plus using global logical constraints for making the inference. A declarative version, Saul, that supported relational and graph-based data modeling and structured perceptron-based learning was proposed in [Kordjamshidi *et al.*, 2015] and its applicability was shown in various domains [Kordjamshidi *et al.*, 2016; Kordjamshidi *et al.*, 2017]. Saul is based on Scala and used Java as backend. This survey [Kordjamshidi *et al.*, 2018] shows the context of Declarative learning-based programming with respect to statistical relational learning, probabilistic programming, probabilistic logical programming, and other related paradigms [Domingos and Richardson, 2004; De Raedt *et al.*, 2007]. This paper is following the same line

of work and takes a step towards declarative learning-based programming by exploiting ontologies of the domain that are available in standard ontology languages.

Sophisticated domain knowledge is required to build learning-based programs to interact with multi-modal data in complex application setting like social media [Guo *et al.*, 2016]. Ontology provides structural knowledge in an application domain that is needed in machine learning models. They have been used in machine learning as a source of background knowledge particularly in biomedical applications for the simpler case of taxonomies [Min *et al.*, 2016; Magumba *et al.*, 2018]. Deep learning models, exploit the ontology as a guidance for attention in this work [Raaijmakers and Brewster, 2018]. Ontologies have been also used in [Wang *et al.*, 2016] to guide the architecture design of the deep restricted Boltzmann machines (DRBM), as well as to assist in their training and validation processes. However, in all of the previous work, to our knowledge, the specific ontological information has been hard encoded into the learning models per case. There is no generic framework to enable the usage of arbitrary ontologies in machine learning in a principled way.

Our pipeline is not intentionally designed to be a computational toolbox. We connect the computations to libraries like PyTorch, TensorFlow, and/or MXNet[6]. The the learner interface provides the flexibility to use various backends. The option is left to the machine learning programmers allowing them to provide whatever underlining computation to the properties of the concepts. A set of other tools with an orthogonal functionality are the tools that are used to process raw data and provide some higher level libraries that help us to put raw data in some data structures, in our work we use NLP tool in our showcase example. Particularly, we use AllenNLP[7] [Gardner *et al.*, 2018] in this study. Basic building blocks like sophisticated tools to extract language features, common neural network layers, and optimizers, configurable command-line usage, are provided out-of-box. However, the proposed pipeline focuses on translating the knowledge implied by ontology to links of the model, loss function, and constraints in inference. This procedure usually needs to be designed carefully with knowledge from both the application domain as well as machine learning.

In contrast to the graph networks [Zhou *et al.*, 2018] which exploit the given graph structure of the instances, the ontologies represent higher level knowledge that can help in generating a graph per instance. Moreover, the ontology can be used as a source of global constraints over the decisions that are made over the data graphs. In the recent GraphNets tool [Battaglia *et al.*, 2018], they represent and predict the nodes, edges, and properties (local and global) of graphs in neural networks. Instance graphs are the input to neural networks in GraphNets formulations. However, our proposed pipeline does not take instance graphs as input to the network. A network is generated, trained, and constrained, based on ontology graphs. In other words, the GraphNets take instance graphs as data while our pipeline takes ontology graphs as the problem to be solved as well as the key to solve it. As the ontology-based reasoning component, we exploit the off-the-shelf linear programming optimization tools. However, there is a recent line of work to train deep networks that can replace the ontological reasoning [Hohenecker and Lukasiewicz, 2018]. This is an orthogonal direction. Our future goal is to exploit heterogeneous reasoning components such as logical, probabilistic inference or constraint optimization models and plug them into the procedure of training rather than replacing all existing formalisms with deep neural networks.

[6]https://mxnet.apache.org/

[7]https://allennlp.org/

4 Conclusion

Ontologies are a rich source of domain knowledge that can be used in machine learning models in several ways. The main idea of this paper is to investigate the compilation of the existing ontologies expressed in standard languages such as OWL into learning based programs. We generate a python program that declares a relational graph and connects the concepts and relationships in the ontologies with sensors and learners. The training targets are inferred automatically and used in loss functions. For the prediction of the target values including entity and relationship labels the learners make the best predictions that are consistent with ontological constraints. The proposed pipeline bridges the gap between the application domain described by ontologies and the learning-based programs.

References

[Battaglia *et al.*, 2018] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018.

[Chang *et al.*, 2012] Ming-Wei Chang, Lev Ratinov, and Dan Roth. Structured learning with constrained conditional models. *Machine Learning*, 88(3):399–431, 6 2012.

[De Raedt *et al.*, 2007] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: a probabilistic Prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2468–2473. AAAI Press, 2007.

[Domingos and Richardson, 2004] Perdo Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In *ICML'04 Workshop on Statistical Relational Learning and its Connections to Other Fields*, pages 49–54, 2004.

[Gardner *et al.*, 2018] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu,

Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. In *NLP OSS workshop, ACL 2018*, 2018.

[Guo *et al.*, 2016] Quan Guo, Jia Jia, Guangyao Shen, Lei Zhang, Lianhong Cai, and Zhang Yi. Learning robust uniform features for cross-media social data by using cross autoencoders. *Knowledge-Based Systems*, 102:64–75, 2016.

[Hohenecker and Lukasiewicz, 2018] Patrick Hohenecker and Thomas Lukasiewicz. Ontology reasoning with deep neural networks. *CoRR*, abs/1808.07980, 2018.

[Kordjamshidi *et al.*, 2015] Parisa Kordjamshidi, Hao Wu, and Dan Roth. Saul: Towards declarative learning based programming. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 7 2015.

[Kordjamshidi *et al.*, 2016] Parisa Kordjamshidi, Daniel Khashabi, Christos Christodoulopoulos, Bhargav Mangipudi, Sameer Singh, and Dan Roth. Better call saul: Flexible programming for learning and inference in NLP. In *Proc. of the International Conference on Computational Linguistics (COLING)*, 2016.

[Kordjamshidi *et al.*, 2017] Parisa Kordjamshidi, Sameer Singh, Daniel Khashabi, Christos Christodoulopoulos, Mark Summons, Saurabh Sinha, and Dan Roth. Relational learning and feature extraction by querying over heterogeneous information networks. In *Seventh International Workshop on Statistical Relational AI (StarAI)*, 2017.

[Kordjamshidi *et al.*, 2018] Parisa Kordjamshidi, Dan Roth, and Kristian Kersting. Systems AI: A declarative learning based programming perspective. In *The 27th International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[Magumba *et al.*, 2018] Mark Abraham Magumba, Peter Nabende, and Ernest Mwebaze. Ontology boosted deep learning for disease name extraction from twitter messages. *J. Big Data*, 5:31, 2018.

[Min *et al.*, 2016] H. Min, H. Mobahi, K.and Krasniqi I. Vukomanovic, S.and Irvin, S. Avramovic, and J. Wojtusiak. Applying an ontology-guided machine learning methodology to SEER-MHOS dataset. *Bio-Ontologies SIG*, 2016.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[Raaijmakers and Brewster, 2018] Stephan Raaijmakers and Christopher Brewster. Exploiting ontologies for deep learning: A case for sentiment mining. In *Proceedings of the Posters and Demos Track of the 14th International Conference on Semantic Systems co-located with the 14th International Conference on Semantic Systems (SEMANTiCS 2018), Vienna, Austria, September 10-13, 2018.*, 2018.

[Rizzolo and Roth, 2010] Nicholas Rizzolo and Dan Roth. Learning based java for rapid development of NLP systems. In *Proc. of the International Conference on Language Resources and Evaluation (LREC)*, Valletta, Malta, 5 2010.

[Rizzolo, 2011] Nicholas Rizzolo. *Learning Based Programming*. PhD thesis, 2011.

[Roth and Yih, 2007] D. Roth and W. Yih. Global inference for entity and relation identification via a linear programming formulation. 2007.

[Roth, 2005] Dan Roth. Learning based programming. *Innovations in Machine Learning: Theory and Applications*, 2005.

[Wang *et al.*, 2016] Hao Wang, Dejing Dou, and Daniel Lowd. Ontology-based deep restricted boltzmann machine. In *Database and Expert Systems Applications - 27th International Conference, DEXA 2016, Porto, Portugal, September 5-8, 2016, Proceedings, Part I*, pages 431–445, 2016.

[Zhang *et al.*, 2016] Xiao Zhang, Maria Leonor Pacheco, Chang Li, and Dan Goldwasser. Introducing DRAIL - a step towards declarative deep relational learning. In *SPNLP@EMNLP*, 2016.

[Zhou *et al.*, 2018] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018.