Logical Rule Extraction for KG Completion

Owen Fish

May 25, 2020

Abstract

Due to the incredible size of fact-based knowledge bases, missing information is very common. In this paper, we examine a way to artificially complete a knowledge base using probabilistic reasoning to extract logical rules over facts. Our model examines the path from node u to node v to determine if nodes u and v should be connected, and if so, what type of connection is appropriate. We fit a RandomForestClassifier to this problem, and we obtained results on par with state-of-the-art methods in KG completion
Keywords: *Knowledge Graph, Logical Rule Extraction*

1 Introduction

Knowledge graphs are strong devices for storing factual information. Similar to a typical graph data structure, KG's consist of a set of vertices (entities) and edges (relationships). Thus, a knowledge graph can store facts as relationships between entities. A perfect knowledge graph (or equivalenty, knowledge *base*) would have a myriad of practical applications for artificial intelligence-related problems including, but not limited to, web search, chat bots, or fraud detection. However, the power of a large-scale knowledge base can also be its main problem: size. With an astronomically large number of entities, and even larger number of facts, it is a nontrivial task to even populate a graph representative of basic human knowledge. To that end, the use of auto-completion algorithms is very common for large-scale knowledge bases.

A fair deal of research has been conducted with regards to populating a knowledge graph through the parsing of web-pages Zhao et al. (2017). Such methods may include semantic analysis of various websites. For example, if an article on a trusted website has the headline: "William & Mary star player Nathan Knight expected to play in NBA," a reasonable prediction is that the entity ”Nathan Knight,” and the entity ”William & Mary Basketball” are linked in the true knowledge base, and a corresponding link should then be placed in our graph. However, this level of computation may not be necessary for the completion of the entire graph. In fact, it is often the case that while a subset of facts are standalone links between entities, another subset of all facts can be reasoned simply by common sense. For example, it is not entirely obvious that ”Nathan Knight” is linked to ”William & Mary Basketball,” but once we know that fact, a link should be placed from ”Nathan Knight” to the ”William & Mary” entity, given the fact that a player on a college team must be enrolled at the school. That is, in this case, the relationship between the links ”playerOnTeam” and ”teamAtSchool” implies the link ”student.”

The primary focus of this paper is on knowledge graph completion methods, with regards to common sense reasoning. Previous research into KG completion methods have revolved around the idea of an embedding of nodes and relationships in vector space. *Learning structured embeddings of knowledge bases* worked on creating an embedding of nodes in vector space to allow the practitioner to infer relationships between nodes by examining transformations of those nodes in that vector space (Bordes, Weston, Collobert, & Bengio, 2011). *Learning entity and relation embeddings for knowledge graph completion* expands on this idea by first transforming each node embedding to a relationship-specific vector space before examining transformations of those nodes in those vector spaces (Lin, Liu, Sun, Liu, & Zhu, 2015). Another extension of Bordes et al, *Reasoning with neural tensor networks for knowledge base completion* aims to complete a knowledge base by training a neural tensor to classify entities e_1, e_2 as having relationship of type R based on the embeddings of nodes e_1, e_2 (Socher, Chen, Manning, & Ng, 2013). That is, the neural tensor makes predictions based on similar entity-entity pairs of a certain relationship type. Using a neural tensor allows the predictions given from Sorcher et al to model a more complex set of interactions than the linear transformations given by Bordes et al and Lin et al. The most similar work to the logical methods in this paper is given by *Entity Context and Relational Paths for Knowledge Graph Completion* (Wang, Ren, & Leskovec, 2020). The authors attempt to predict (e_1, R, e_2) by first learning the logical rules associated with the various *paths* from e_1 to e_2. They then combine these rules with the contextual embeddings of the source and sink nodes to make a prediction about (e_1, R, e_2). This framework can accurately model the relationship between logical rules and the context around them: some rules only apply in certain situations. However, covariatizing the node embeddings can act as a safety net for model performance and, ultimately, inhibit the learning process of the logic-based rules, if included. To ensure lack of model regression to simply training on the embeddings, we do not include them in our model.

2 Problem Formulation

Let $G = (V, E)$ be our current Knowledge Graph, and R be the set of relationships. Note that $None \in R$, where $(e_1, None, e_2)$ indicates that entities e_1, e_2 are not connected by any link in our graph. It is important to note that the set of relationships R is finite. Thus, we seek to build a classifier that estimates

$$P((e_1, R_i, e_2)|\mathscr{P}(e_1, e_2)) \qquad \text{For all } R_i \in R$$

, where $\mathscr{P}(e_1, e_2)$ are the ordered relationships $(r_1, r_2, ..., r_k)$ along a path from e_1 to e_2. For this project, the path is constructed randomly, with the only parameter the length, $k = 2$ in this case.

With a classification problem, as we have here, a common loss function is categorical cross entropy.

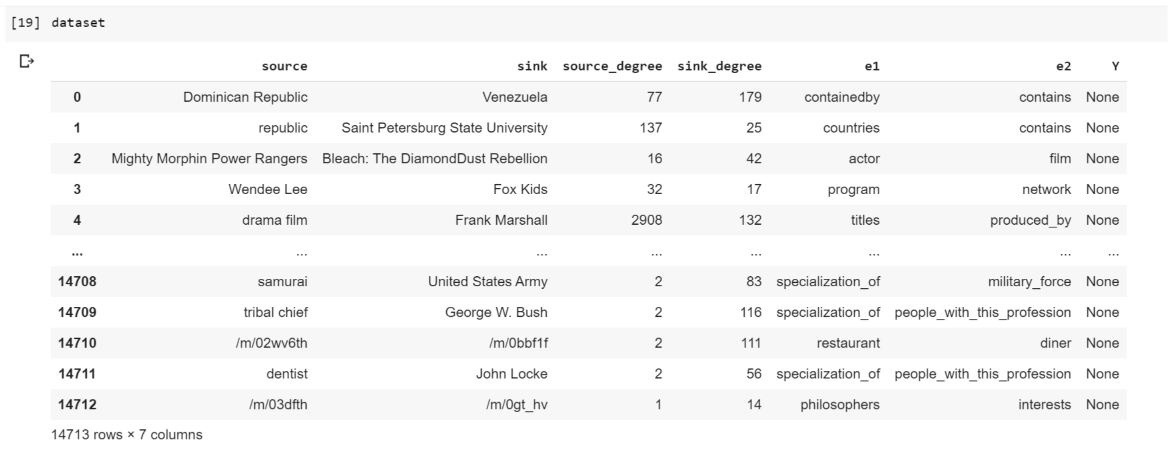
3 Data & Methodology

Our data set is derived from the Freebase FB15K open-source Knowledge Graph. We used the link provided by Github user villmow. We first loaded the set of triplets (e_1, R_1, e_2) into a NetworkX Directed MultiGraph using Python. From there, we randomly sampled 15,000 paths of length $k = 2$ using the code in the attached Jupyter notebook DatasetCreation.ipynb. We then looked if the graph had an edge to connect the endpoints of the paths; this was our ground truth data. As is the case when training models with categorical variables, we used one-hot-encoding to encode the categorical relationship variables into numeric vectors. Such encoding transform a relationship of type i into a vector of all zeros, except a 1 in position i. Because all the possible relationship types are in the set R, we used a single encoder over all $r \in R$, and then applied it to each entry of the data set. Thus, the same link will have the same encoding no matter what position it is in in our data set. We also included node degrees for both the source and sink nodes, as this value gives a proxy for node "popularity," and it stands to reason that the more neighbors a given node has, the more likely it is to be connected to any particular node. We tracked the names of the source and sink nodes, but did not feed them to the classifier.

The above screenshot shows a few rows of our data set. One thing to notice is that the distribution of ground truth relationships is not approximately uniform, which is a desirable attribute of multiclass classification problems. In fact, we have severly imbalanced data, biased toward no relationship between nodes u, v.

$$Y \not\sim U_R(x)$$

where the random variable Y is given by the ground truth relationships between nodes u, v, and $U_R(x)$ denotes a discreet uniform distribution over the elements of R. This imbalance is displayed in the below chart, which shows the counts of each relationship type in the ground truth column of our data set.



```
[19] dataset
```

	source	sink	source_degree	sink_degree	e1	e2	Y
0	Dominican Republic	Venezuela	77	179	containedby	contains	None
1	republic	Saint Petersburg State University	137	25	countries	contains	None
2	Mighty Morphin Power Rangers	Bleach: The DiamondDust Rebellion	16	42	actor	film	None
3	Wendee Lee	Fox Kids	32	17	program	network	None
4	drama film	Frank Marshall	2908	132	titles	produced_by	None
...	...	...	...	...	...	...	...
14708	samurai	United States Army	2	83	specialization_of	military_force	None
14709	tribal chief	George W. Bush	2	116	specialization_of	people_with_this_profession	None
14710	/m/02wv6th	/m/0bbf1f	2	111	restaurant	diner	None
14711	dentist	John Locke	2	56	specialization_of	people_with_this_profession	None
14712	/m/03dfth	/m/0gt_hv	1	14	philosophers	interests	None

14713 rows × 7 columns

Figure 1: Rows of data used

To address this problem, we took two precautions:

3.1 Over-Sampling

A common method to combat class imbalance, over-sampling is the procedure of randomly replicating samples in an under-represented class until the classes have roughly the same amount of representation in the data set. The Python library Scikit-Learn provides a RandomOverSampler object to over-sample the relevant data and stitch together the final dataset.

As is clear by the above figure, the ground truth distribution was heavily skewed toward ”None.” The following figure shows the distribution in the training set after randomly over sampling. In this figure, each bar represents the count of records in the training set that have ground truth value of that type. The counts all are approximately equal, and this will help the training process learn.

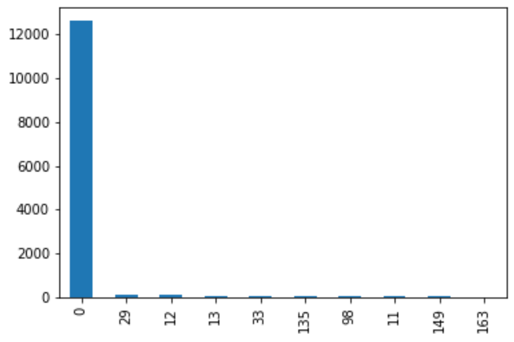


Figure 2: Ground Truth Distribution

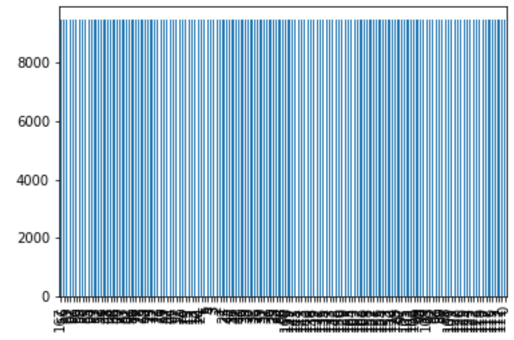


Figure 3: Ground Truth after over-sampling

3.2 Model Selection

The main issue with imbalanced data is that a learning algorithm could exploit the imbalance to give accurate predictions without learning anything. If the training set has 90% of its records of a certain class, a constant classifier that always predicts that class would have a training accuracy of 90%. This laziness occurs to different degrees with different models. Tree-based methods, specifically ensembled trees, are known to perform relatively well on imbalanced data (Rodríguez, Díez-Pastor, & García-Osorio, 2011). For this reason, we used a RandomForest classifier to model this problem.

3.3 Metric

As simple misclassification rate can give a misleading view of model performance in the face of imbalanced data, it is in our best interest to verify the misclassification rate with a second performance metric. The Receiver Operating Characteristic Curve (ROC) is unaffected by imbalanced data, and thus a good choice for a performance metric Jeni, Cohn, and De La Torre (2013).

4 Results

When we fit a Random Forest classifier to our over-sampled training set (with $n = 20$ trees), we obtained a test accuracy of 92%. So, our model was able to correctly classify whether or not a link in the graph exists over 90% of the time.

Below is the associated ROC curves for each relationship type. The curves show the tradeoff between the True positive rate and false positive rate whith regards to a specific relationship type. As the graph shows, some relationships had constant, or nearly constant, true positive rates, while some had closer to a linear relationship with the false positive rate. However, for all of the possible classes, the entire ROC curve lived above the perfectly linear relationship given by the black dotted line. Since our test set did not have complete representation of each relationship type, the ROC is not defined for each class, and only the classes with a defined curve are shown.

These results are relatively consistent with those found in similar literature. TransE, the method introduced in (Bordes et al., 2011) obtained a 96% accuracy on the same data set, while PathCon, the method introduced in (Wang et al., 2020) obtained a 98.4% accuracy.

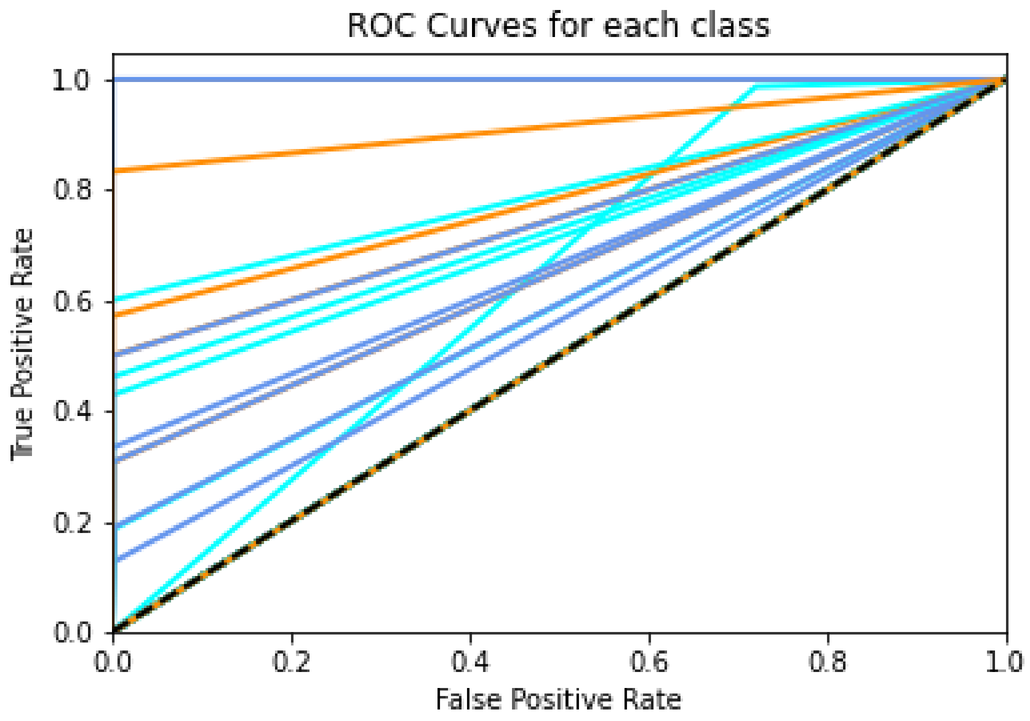


Figure 4: ROC Curve for relationship types with representation in test set

5 Conclusions and Future Work

The realization of logical rules are evident in basic human life. There seemingly exists a set of metadata that govern a significant portion of facts themselves. This set can include when and where facts are transitive; that is, if entity a is connected to entity b, and entity b is connected to entity c, it is obvious that entity a should be connected to entity c. If you let the relationship in question be whether or not entities a and b are siblings, then it is clear that the sibling relationship is transitive: my brother's brother is also my brother. However, transitivity does not follow for, say, the relationship *enemy*. The cliche does not specify that the enemy of my enemy is also my enemy. This set of metadata is so blatantly obvious to any human of a reasonable age, but very difficult to enumerate. So difficult, in fact, that it comes into question if this extraction of logical rules is even worth the trouble, or if another method for understanding relationships among entities is a safer bet. After all, the problem at hand is artificial completion of the knowledge base, not rule extraction in and of itself. However, this mindset does not fully take into account the benefits of learning rules in a knowledge graph. As my KG becomes bigger and bigger, it becomes more and more important for the developer to find ways to mitigate that size. Learning logical rules can allow the KG to discard storage of derived relationships in favor of transformation at program run-time, and, in a way, orthogonalize the knowledge base. Not to mention the uses of logical extraction with regards to KG completion.

Significant research has been conducted on logical rule extraction for other problems, and knowledge bases would be a very interesting application.

The model proposed in this paper has several upgrades available. First, in the future, we could obtain our dataset in any of a number of different manners. In this paper, we obtained the data set by finding a random path of fixed length in our knowledge graph. This method not only gave us an imbalanced data set, but we are also not making any attempt to uncover a principal set of relationships. Using a more defined sampling procedure could end up giving us more meaningful data. Another extension of this model could be to use a Recurrent Neural Network (RNN) model rather than a random forest. A RNN allows for arbitrary path lengths, which is more extensible to the real data set. Also, this would allow us to take multiple paths from u to v into consideration, which may be useful.

References

Bordes, A., Weston, J., Collobert, R., & Bengio, Y. (2011). Learning structured embeddings of knowledge bases. In *Twenty-fifth aaai conference on artificial intelligence.*

Jeni, L. A., Cohn, J. F., & De La Torre, F. (2013). Facing imbalanced data–recommendations for the use of performance metrics. In *2013 humaine association conference on affective computing and intelligent interaction* (pp. 245–251).

Lin, Y., Liu, Z., Sun, M., Liu, Y., & Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth aaai conference on artificial intelligence.*

Rodríguez, J. J., Díez-Pastor, J. F., & García-Osorio, C. (2011). Ensembles of decision trees for imbalanced data. In *International workshop on multiple classifier systems* (pp. 76–85).

Socher, R., Chen, D., Manning, C. D., & Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems* (pp. 926–934).

Wang, H., Ren, H., & Leskovec, J. (2020). Entity context and relational paths for knowledge graph completion. *arXiv preprint arXiv:2002.06757.*

Zhao, X., Xing, Z., Kabir, M. A., Sawada, N., Li, J., & Lin, S. (2017). Hdskg: Harvesting domain specific knowledge graph from content of webpages. In *2017 ieee 24th international conference on software analysis, evolution and reengineering (saner)* (p. 56-67).