Predicting Invariant Nodes in Large Scale Semantic Graphs

Damian Barsotti Ma

Martin Ariel Dominguez

Pablo Ariel Duboue

FaMAF-UNC / Cordoba, Argentina

damian, mdoming, pablod@famaf.unc.edu.ar

Abstract

We are interested in understanding and predicting how large knowledge graphs change over time. An important subproblem is predicting which nodes within the graph won't have any edges deleted or changed (what we call add-only nodes). Predicting add-only nodes correctly has practical importance, as such nodes can then be cached or represented using a more efficient data structure. This paper presents a logistic regression approach using attribute-values as features that achieves 95%+ precision on DBpedia yearly changes trained using Apache Spark. It concludes by outlining how we plan to use these models for Natural Language Generation.

1 Introduction

We are interested in understanding and predicting how large knowledge graphs change over time. An important subproblem is predicting which nodes within the graph won't have any edges deleted or changed (what we call add-only nodes) or undergo any changes at all (what we call constant nodes). Predicting add-only nodes correctly has practical importance, as such nodes can then be cached or represented using a more efficient data structure. In this paper we show a logistic regression approach using attribute-values as features that achieves 95%+ precision on DBpedia¹ yearly changes, as trained using Apache Spark. We conclude by outlining how we plan to use these models for Natural Language Generation.

Definition. Given a multigraph G_0 with named edges such that each source node S is linked through an edge labeled V to a target node O,

which we will call a *triple* $\langle S, V, O \rangle$, we will say a given node S is an *add-only node* if in a next version (G₁) of the multigraph, all triples starting on S in G₀ are also in G₁. That is, S is *add* - *only* iff :

 $\forall v, o \mid \langle S, v, o \rangle \in G_0 \Rightarrow \langle S, v, o \rangle \in G_1$

This type of nodes can be efficiently represented as static information, for example by leveraging large scale perfect hashes (Botelho and Ziviani, 2007).

Our intuition is that in large scale semantic graphs holding an imperfect representation of the real world, there will be two types of changes, (1) model enhancements, where the truth about the world is better captured by the model and (2) model corrections, where the world has changed and the model is updated. Updates of the first type result in new information added to the graph, without modifying existing data. Finding such nodes is the objective of our work.

This work is structured as follows. In the next section we summarize related work. In Section 2 we discuss DBpedia, the semantic graph we used for our experiments. Our methods and result follows, closing with a discussion of our intended application in Natural Language Generation.

2 Related Work

Mining graphs for nodes with special properties is not new to Big Data mining (Drury et al., 2015). With the development DBpedia much research has been devoted to exploiting this resource in AI tasks as well as to model its changes. For example, there is research on modeling DBpedia's currency (Rula et al., 2014), that is, the age of the data in it and the speed at which those changes can be captured by any system. Although currency could

¹http://dbpedia.org

be computed based on the modification/creation dates of the resources, this information is not always present in Wikipedia pages. To overcome this, the authors propose a model to estimate currency combining information from the original related pages and a couple of currency metrics measuring the speed of retrieval by a system and basic currency or timestamp. Their experiments suggest that entities with high system currency are associated with more complete DBpedia resources and entities with low system currency appear associated with Wikipedia pages that are not easily tractable (or that "could not provide real world information" according with the authors). While both the authors and us look into changes in DBpedia, we are interested in changes that for the most part do not result from changes in the real world, as Lehman and others are interested.

The need to account for changes in ontologies has long been acknowledged, given that they may not be useful in real world applications if the representation of the knowledge they contain is outdated. Eder and Koncilia (Eder and Koncilia, 2004) present a formalism to represent ontologies as graphs that contain a time model including time intervals and valid times for concepts. They base their formalism on techniques developed for temporal databases, namely the versioning of databases instead of their evolution and they provide some guidelines about its possible implementation. Our work can be used to improve the internal representation of such temporal databases (Cheng et al., 2014).

Another source of ontology transformation is spatiotemporal changes. Dealing with spatial changes in historical data (or over time series) is crucial for some NLP tasks, such as information retrieval (Kauppinen and Hyvnen, 2007). In their case, the authors deal with the evolution of the ontology's underlying domain instead of its versioning or evolution due to developments or refinements. Their main result is the definition of partial overlaps between concepts in a given time series, which was applied to build a Finnish Temporal Region Ontology, showing promising results.

Finally, we see parallelisms between change tracking in other large graphs: object graphs in garbage collection systems. State of the art garbage collection will single out objects that survive multiple garbage collections (Stefanović et al., 1999) and stop considering them for collection. It is this type of optimizations that we expect detection of invariable nodes will help semantic graphs updates.

3 Data

As a large scale naturally occurring knowledge graph with a rich update history, we use DBpedia, a knowledge graph derived from the Wikipedia collaborative encyclopedia started in January 2001 at present containing over 37 million articles in 284 languages.

Given that the content in Wikipedia pages is stored in a structured way, it is possible to extract and organize it in an ontology-like manner as implemented in the DBpedia community project. This is accomplished by mapping Wikipedia infoboxes from each page to a curated shared ontology that contains 529 classes and around 2,300 different properties. DBpedia contains the knowledge from 111 different language editions of Wikipedia and, for English the knowledge base consists of more than 400 million facts describing 3.7 million things (Lehmann et al., 2015). A noble feature of this resource is that it is freely available to download in the form of *dumps* or it can be consulted using specific tools developed to query it.

These dumps contain the information in a language called Resource Description Framework (RDF) (Lassila et al., 1998). The WWW Consortium (W3C) has developed RDF to encode the knowledge present in web pages, so that it is comprehensible and exploitable by agents during any information search. RDF is based on the concept of making statements about (web) resources using expressions in the subject-predicate-object form. These expressions are known as triples, where the subject denotes the resource being described, the predicate denotes a characteristic of the subject and describes the relation between the subject and the object. A collection of such RDF declarations can be formally represented as a labeled directed multi-graph, naturally appropriate to represent ontologies.

Table 1 shows the different years employed in this work. The DBpedia project obtains its data through a series of scripts run over Wikipedia, which on itself is a user-generated resource. Changes to the DBpedia scripts or to Wikipedia itself sometimes result in dramatic differences from one year to the next. Besides the overall sizes, what is relevant to this work is the total number

Table 1: Data sizes.

Version	# Nodes	# Links
2010-3.6	1,668,503	19,969,604
2011-3.7	1,831,219	26,770,236
2012-3.8	2,350,907	33,742,028
2013-3.9	3,243,478	41,804,713
2014	4,218,628	61,481,509
2015-04	4,080,388	37,791,134
2015-10	4,995,949	40,617,978
2016-04	5,109,879	40,407,197

Table 2: Percentage of entities which remains constant or add-only calculated between two consecutive versions of DBPedia.

Consecutive versions		% Const.	% Add.
2010-3.6	2011-3.7	9.71	45.73
2011-3.7	2012-3.8	30.28	65.51
2012-3.8	2013-3.9	38.72	76.79
2013-3.9	2014	16.61	49.32
2014	2015-04	14.01	29.01
2015-04	2015-10	3.76	20.54
2015-10	2016-04	83.63	90.52

of additions and deletions, shown in Table 2.

4 Methods

Our prediction system is implemented using Apache Spark² using its Logistic Regression package. In our algorithm the feature vector itself is comprised of binary features indicating whether or not a given relation object holds for the subject in OLD; that is, we do not look at whether the $\stackrel{V_i}{\rightarrow}O_i$ have changed, just their existence in OLD. The class is, given a node in subject position, S:

add-only: $\{(V_i, O_i)\}_{OLD} \subseteq \{(V_i, O_i)\}_{NEW}$ constant: $\{(V_i, O_i)\}_{OLD} = \{(V_i, O_i)\}_{NEW}$

The feature vector underneath has a dimension of $||V|| \times ||O||$, a potentially very large number given the millions of values in O. We leverage Apache Spark Mlib pipelines to filter out this extremely large feature vector to the top million entries.

Figure 1 shows a small example of feature and class extraction. A four node graph OLD evolves into a five node graph NEW. The classes for each node are computed over OLD, using three binary features.

5 Results

Using the machine learning method described in the previous section we took three consecutive year, $G_{y_1},G_{y_2},G_{y_3},$ built a model M on G_{y_1} \rightarrow G_{y_2} , apply M on G_{y_2} , obtaining G'_{y_3} and evaluate it by comparing it to G_{y_3} . Table 3 shows our results. We can see that for pairs close in size (from Table 1) we obtain a precision close to 90% with recall ranging from 20% to 58% (and F-measure as high as 66%). As our numbers were obtained by optimizing F1 on a binary classification problem, precision and recall are dual and treated as identical quantities by the optimizer (this can be easily seen from a confusion table). The rows marked with an asterisk in the table were inverted in our experiments. The low numbers for the last row in the table can be attributed to ontological re-estructuring on Wikipedia/DBpedia on 2015-04/10 period (second to last row on Table 2) were few entities remained constant through 2015.

From the table we can also see that detecting constant nodes, on the other hand, it is a much more difficult task that might be ill-defined given the nature of the updates discussed in the intro. Table 4 shows some examples of correctly and incorrectly predicted nodes.

5.1 Discussion

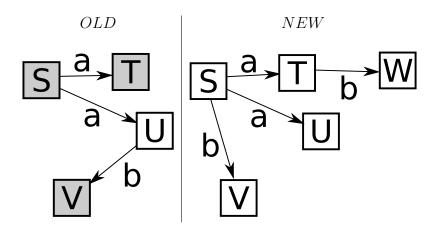
How useful are these results? For the task we have in mind, building statistically plausible future versions of existing semantic graphs for the purpose of testing Natural Language Generation algorithms (Duboue et al., 2016), successfully predicting add-only nodes help us immediately with the performance of the prediction system. Our high precision results will then carry over to direct improvements on our full system: if our system has an error rate of 30% and there are 25% of add-only nodes, our current system will reduce error by up to 12% (in the case of 50% recall).

Another case is for maintaining the data. The add-only nodes and relations can be pre-cached using more efficient data structures such as perfect hashes (Botelho and Ziviani, 2007).

6 Conclusions and Future Work

In Natural Language Generation (Reiter and Dale, 2000), Referring Expressions Generation (REG), is the task that, given an entity (the **referent**) and a set of competing entities (the **set of distractors**), involves creating a mention to the referent so that,

²https://spark.apache.org/



Node	Features	Tar	get
S	a=T, a=U	add-only	\neg constant
Т	Ø	add-only	\neg constant
U	b=V	\neg add-only	\neg constant
V	Ø	add-only	constant

Figure 1: Feature generation from an ontology OLD and NEW. In this example most nodes are add-only (shaded in OLD), only U loses a relation and it is thus not add-only.

Tra	ain	Eval		Sys	tem	
Source	Target	Target	Add-o	only	Const	ant
			Precision	Recall	Precision	Recall
2010-3.6	2011-3.7	2012-3.8	0.560	0.579	0.704	0.887
2011-3.7	2012-3.8	2013-3.9	0.448	0.444	0.658	0.569
2012-3.8	2013-3.9	2014	0.916	0.224	0.890	0.472
2013-3.9	2014	2015-04	0.971	0.506	0.965	0.770
2014	2015-04	2015-10	0.989	0.650	0.971	0.820
2015-04	2015-10	2016-04	0.945	0.196	0.908	0.068

Table 3: Training in two consecutive years and evaluating on a third. Training maximizing F1.

Table 4: Example predictions and mispredictions, using $2015-04 \rightarrow 2015-10$ as training and tested on 2016-04.

Correctly predicted add-only		
Iasi_Botanical_Garden	constant	
USS_Breakwater_(SP-681)	constant	
Interborough_Handicap	constant	
Thode_Island	added archipelago→Marshall_Archipelago	
Colonel_Reeves_Stakes	added location	
	added location	

inconteent predicted as add-only	Incorrec	y predicted as ad	d-only
----------------------------------	----------	-------------------	--------

Beverly_Hills_Handicap	disappears due to name change
First_Ward_Park	disappears due to name change
2012_Shonan_Bellmare_season	changes league \rightarrow 2012_J. League_Division_2
	to league \rightarrow 2012_J.League_Division_2

in the eyes of the reader, it is clearly distinguishable from any other entity in the set of distractors. Therefore REG algorithms are expected to select attributes that unambiguously identify an entity with respect to a set of distractors.

Our current work is part of a plan to simulate natural perturbations on the data in order to find the conditions on which REG algorithms start to fail (for example, a simulated DBpedia 25 years in the future).

In previous work we explored the robustness for the particular case of Referring Expressions Generation (REG) algorithms by means of different versions of an ontology (Duboue et al., 2016).

In (Duboue and Domínguez, 2016) we presented experiments on two types of entities (people and organizations) and using different versions of DBpedia we found that robustness of the tuned algorithm and its parameters do coincide but more work is needed to learn these parameters from data in a generalizable fashion.

We plan to extend the current model with a specific model for additions and deletions using techniques from statistical machine translation (Koehn, 2010) and investigate techniques based on knowledge embedding models (Xie et al., 2017).

Acknowledgments

The authors would like to thank the Secretaria de Ciencia y Tecnica of Cordoba Province for support and the anonymous reviewers for helpful comments and suggestions.

References

- Fabiano C. Botelho and Nivio Ziviani. 2007. External perfect hashing for very large key sets. In Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management. ACM, New York, NY, USA, CIKM '07, pages 653– 662. https://doi.org/10.1145/1321440.1321532.
- S. Cheng, A. Termehchy, and V. Hristidis. 2014. Efficient prediction of difficult keyword queries over databases. *IEEE Transactions on Knowledge and Data Engineering* 26(6):1507–1520. https://doi.org/10.1109/TKDE.2013.140.
- Brett Drury, Jorge Carlos Valverde-Rebaza, and Alneu de Andrade Lopes. 2015. Causation generalization through the identification of equivalent nodes in causal sparse graphs constructed from text using node similarity strategies. In *Proceedings of the 2nd Annual International Symposium on Information Management and Big Data - SIMBig 2015,*

Cusco, Peru, September 2-4, 2015.. pages 58–65. http://ceur-ws.org/Vol-1478/paper6.pdf.

- Pablo Ariel Duboue and Martin Ariel Domínguez. 2016. Using Robustness to Learn to Order Semantic Properties in Referring Expression Generation, Springer International Publishing, Cham, pages 163–174.
- Pablo Ariel Duboue, Martin Ariel Dominguez, and Paula Estrella. 2016. On the robustness of standalone referring expression generation algorithms using rdf data. *WebNLG 2016* page 17.
- Johann Eder and Christian Koncilia. 2004. C.: Modelling changes in ontologies. In *In: Proceedings of* On The Move - Federated Conferences, OTM 2004, Springer (2004) LNCS 3292. pages 662–673.
- Tomi Kauppinen and Eero Hyvnen. 2007. Modeling and reasoning about changes in ontology time series. In *Integrated Series in Information Systems*. Springer-Verlag, pages 319–338.
- Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.
- Ora Lassila, Ralph R. Swick, World Wide, and Web Consortium. 1998. Resource description framework (rdf) model and syntax specification.
- Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal* 6(2):167–195.
- Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.
- Anisa Rula, Luca Panziera, Matteo Palmonari, and Andrea Maurino. 2014. Capturing the currency of db-pedia descriptions and get insight into their validity. In Proceedings of the 5th International Workshop on Consuming Linked Data (COLD 2014) co-located with the 13th International Semantic Web Conference (ISWC 2014), Riva del Garda, Italy, October 20, 2014..
- Darko Stefanović, Kathryn S McKinley, and J Eliot B Moss. 1999. Age-based garbage collection. ACM SIGPLAN Notices 34(10):370–381.
- Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. 2017. An interpretable knowledge transfer model for knowledge base completion. In ACL 2017: Proceedings of the Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, Vancouver, Canada.