Predicting Invariant Nodes in Large Scale Semantic Knowledge Graphs

Damian Barsotti, Martin Ariel Dominguez, and Pablo Ariel Duboue

FaMAF-UNC / Cordoba, Argentina,
{damian,mdoming,pablod}@famaf.unc.edu.ar

Abstract. Understanding and predicting how large scale knowledge graphs change over time has direct implications in software and hardware associated with their maintenance and storage. An important subproblem is predicting invariant nodes, that is, nodes within the graph will not have any edges deleted or changed (add-only nodes) or will not have any edges added or changed (del-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 90%+ precision on DBpedia yearly changes trained using Apache Spark. The paper concludes by outlining how we plan to use these models for evaluating Natural Language Generation algorithms.

1 Introduction

We are interested in understanding and predicting how large scale knowledge graphs change over time. An important subproblem is predicting which nodes within the graph will not have any edges deleted or changed (what we call add-only nodes) or undergo any changes at all (what we call constant nodes) and which ones will not have any edges added or changed (del-only). 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 90%+ precision on DBpedia[1] yearly changes, as trained using Apache Spark. We conclude by outlining how we plan to use these models for Natural Language Generation. Add-only nodes can be efficiently represented as static information, for example by leveraging large scale perfect hashes [3].

Our intuition is that in large scale semantic graphs holding an imperfect representation of the real world, there are two types of changes, (1) **model enhancements,** where the truth about the world is better captured by the model (for example, the model missed any data about the children of an actor and that has been subsequently included) and (2) **model corrections,** where the world has changed and the model is updated (for example, an actress gave birth to a newborn and that gets included in her page). Updates of the first type

[1] http://dbpedia.org

result in new information added to the graph, without modifying existing data. Finding such nodes is the objective of our work.

Our experiments employed eight different versions of DBpedia, a large scale knowledge graph with several million nodes. Our results show that for updates similar in size, models training in past data can help identify add-only nodes in future data. From our experiments, we believe better data in terms of stable extraction scripts and equally spaced regular intervals is needed rather than relying in the data dumps graciously provided by the DBpedia project.

This paper is structured as follows: in the next section we summarize related work. In Section 3 we discuss DBpedia, the semantic graph we used for our experiments, and how we build the datasets for our experiments. In Section 4 we describe the machine learning and other methods used for this work. The result of experiments are described in Section 5. We close 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 [5]. With the development of DBpedia, much research has been devoted to exploiting this resource in AI tasks as well as to model its changes. For example, Lehman and others' work was on modeling DBpedia's currency [18], that is, the age of the data it contains and the speed at which those changes can be captured by any system. Although currency could be computed based on the modification/creation dates of the resources, this information is not always present in Wikipedia pages. To overcome this problem, the authors propose a model to estimate currency combining information from the original related pages and 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 are associated with Wikipedia pages that are not easily tractable (or that "could not provide real world information" according with the authors). While we also 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. Using the nomenclature from the introduction, they focus on model corrections while we are also interested in model enhancements.

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 [9] 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 [4].

Another source of ontology transformation is spatio-temporal changes. Dealing with spatial changes in historical data (or over time series) is crucial for some NLP tasks, such as information retrieval [10]. 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 tracking change in DBpedia and in other large graphs, namely, object graphs in garbage collection systems. State of the art garbage collection will single out objects that survive multiple garbage collections [19] and stop considering them for collection. We expect that it is this type of optimizations where the detection of invariable nodes will help semantic graphs updates.

3 Data

We investigate the task of prediction invariant nodes by using DBpedia, a knowledge graph derived from the Wikipedia collaborative encyclopedia started in January 2001 at present containing over 37 million articles in 284 languages. We selected DBpedia because it is a large scale naturally occurring knowledge graph with a rich update history.

Given that the content in Wikipedia pages contains structured information in their side boxes ("infoboxes"), 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. These mappings are themselves organized as a wiki and updated over time. 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 [13]. 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) [12]. 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.

Next, we formally define this concept.

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 (G_1) of the multigraph, all triples starting on S in G_0 are also in G_1. That is, S is $add-only$ iff :

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

Similarly, S is $del-only$ iff:

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

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 resulting in many nodes to disappear and new nodes to be created (Table 2), in addition to natural changes in Wikipedia. Besides the overall sizes, what is relevant to this work is the total number of additions and deletions, shown in Table 3.

Table 1. Data sizes with percentage increase from previous version.

Version	# Unique Subjects	# Links
2010-3.6	1,638,799	13,608,535
2011-3.7	1,827,598 (111.52%)	17,228,764 (126.60%)
2012-3.8	2,343,007 (128.20%)	20,174,709 (117.09%)
2013-3.9	3,241,132 (138.33%)	25,346,359 (125.63%)
2014	4,172,476 (128.73%)	33,296,974 (131.36%)
2015-04	4,030,472 (96.59%)	32,018,293 (96.15%)
2015-10	4,903,501 (121.66%)	35,436,595 (110.67%)
2016-04	4,954,767 (101.04%)	35,085,378 (99.00%)

Table 2. Percentage of entities that disappear or appear anew in the next version.

Consecutive		Old Subjects	New Subjects
2010-3.6	2011-3.7	13.53%	22.47%
2011-3.7	2012-3.8	19.38%	37.12%
2012-3.8	2013-3.9	8.98%	34.20%
2013-3.9	2014	5.25%	26.39%
2014	2015-04	27.29%	24.72%
2015-04	2015-10	13.77%	29.12%
2015-10	2016-04	4.03%	5.02%

4 Methods

Our prediction system is implemented using Apache Spark[2] using the distributed Logistic Regression package in MLlib [15]. 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 $\langle V_i, O_i \rangle$ 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: $\{(V_i, O_i)\}_{OLD} = \{(V_i, O_i)\}_{NEW}$
del-only: $\{(V_i, O_i)\}_{OLD} \supseteq \{(V_i, O_i)\}_{NEW}$

The full feature vector has a dimension of $\|V\| \times \|O\|$, a potentially very large number given the millions of values in O (for DBpedia $\|V\|$ is in the order of a thousand and $\|O\|$ is larger than the number of unique subjects in Table 1 but of comparable size). We leverage Apache Spark Mlib pipelines to filter out this extremely large feature vector to the top 200,000 entries.

We encode the DBpedia mapping files into Spark native Parquet file format [14], totaling 8.6Gb on disk, for all DBpedia versions used. To compute the class labels efficiently, we profit from Spark SQL join operations [2]: we join the two consecutive versions and keep all unique subjects present on the old version. We then find whether there are any triples in the new version that has been added or deleted. Depending on the existence of such triples, we obtain three binary labels and use them to train three different classifiers (constant, add-only and del-only). The "other" class in Table 3 are the nodes that are negative for all three categories. After training the logistic regression a threshold on its value that maximizes F1 is determined by cross validation on the train data.

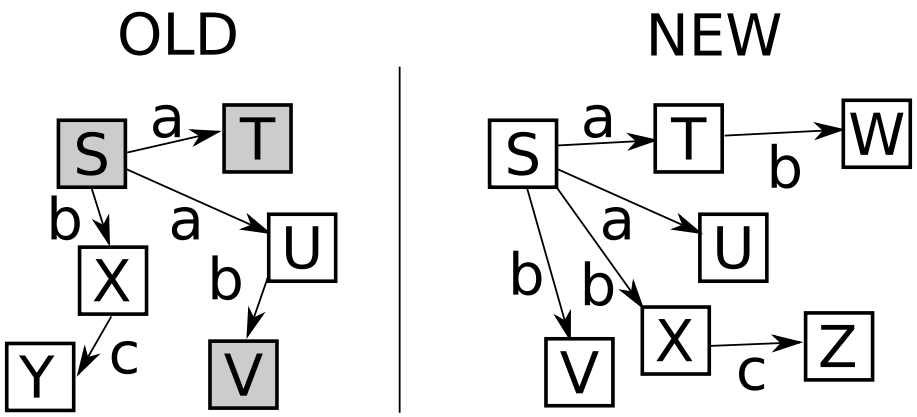
We also performed a drill-down for the errors for entities of different types. We used the types for evaluation but we did not use them for training as we did not want to bias the system unnecessarily but we revisit this decision in the conclusions.

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 4 shows our results which are better understood by comparing them to Table 3 on the same page. Besides G_{y_3}, we also apply M to $G_{y_k} \rightarrow G_{y_{k+1}}$ for all $y_{k+1} > y_3$. All the experiments described here took a week to run using 20Gb of RAM on a

[2] https://spark.apache.org/



Node	Features	Target
S	a=T, a=U	add-only ¬ constant ¬ del-only
T	∅	add-only ¬ constant ¬ del-only
U	b=V	¬ add-only ¬ constant del-only
V	∅	add-only constant del-only
X	c=Y	¬ add-only ¬ constant ¬ del-only
Y	∅	¬ add-only ¬ constant del-only

Fig. 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. Note that W and Z are not considered as they do not exist in OLD.

8-core server. We can see that for pairs close in size (from Table 1) we obtain a precision close to 90% with recall ranging from 42% to 90% (and F-measure as high as 70%). *A priori,* we could expect that a model trained on data closer in time should predict better that one trained farther in time. The table, however, shows that models trained in periods with fewer changes perform better when applied to times with fewer changes and similarly for large changes. That is, in these experiments priors trump recency. Now, the uneven nature of the priors is an artifact of using DBpedia dumps as released by the project, which happen at irregular intervals and contain changes in Wikipedia concurrent with changes in the code used to obtain the ontology. We believe further experimentation on this problem should be done running a fixed version of the DBpedia scripts over Wikipedia historical data directly, a topic we discuss in the conclusions.

The low numbers in the bottom of the table can be attributed to ontological re-structuring on Wikipedia/DBpedia on 2015-04/10 period (second to last row on Table 3) were few entities remained constant through 2015.

From the table we can also see that detecting constant nodes, on the other hand, is a much more difficult task given the fact that Wikipedia is always growing and no model is ever 100% accurate. As such, model enhancements are always to be expected. Del-only results are much difficult to interpret because they contain an extra class of entities, which dominate the numbers (the "old subjects" column in Table 2, or the difference between del-only and proper del-only plus constant in Table 3): the entities that got removed from one version to another. While the numbers themselves are higher than add-only, we do not believe them to be comparable. The large number of deletions can be attributed to changes in the DBpedia scripts (from example, renaming "Chicago (City)" to "Chicago, Illinois") that result in large scale entity deletion and creation.

Table 8 shows some examples of correctly and incorrectly predicted nodes.

5.1 Type Analysis

We also took all types present in at least six versions and combined all the predictions across all triples of consecutive years. We then analyze the behavior of our approach on the top most frequent 50 types, focusing on the ones above and below average (Table 5). From the 22 types below average (Table 6), 17 are related to people or locations. The table also highlights that the numbers vary substantially from year to year although certain years are clearly worse across the board: 2012-2013 look reasonably good while subsequent years experimented plenty of ontological changes due to the introduction of the OWL ontology [1] and the launch of the Wikidata project [21]. The analysis for relatively good types (Table 7) shows more stability per year. The types in both lists make conceptual sense: insects and bodies of water will usually have missing information added over time but data is seldom modified. On other hand, soccer players have statistics continually updated and they change teams fairly often.

Table 3. Percentage of entities which remains constant or add-only calculated between two consecutive versions of DBpedia. Here, "proper" means add-only that are not constant and del-only that are neither constant nor disappeared in the next version.

Consecutive versions		Constant	Add-only		Del-only		Other
				Proper		Proper	
2010-3.6	2011-3.7	12.14%	32.13%	19.98%	30.11%	4.43%	49.89%
		199,047	526,550	327,503	493,556	72,646	817,740
2011-3.7	2012-3.8	44.69%	49.10%	4.40%	65.91%	1.83%	29.67%
		816,830	897,386	80,556	1,204,727	33,558	542,315
2012-3.8	2013-3.9	57.04%	65.80%	8.75%	72.61%	6.59%	18.62%
		1,336,488	1,541,727	205,239	1,701,442	154,523	436,326
2013-3.9	2014	22.24%	32.09%	9.84%	33.11%	5.61%	57.04%
		721,117	1,040,162	319,045	1,073,183	181,899	1,848,904
2014	2015-04	13.77%	16.08%	2.31%	42.50%	1.43%	55.18%
		574,767	671,268	96,501	1,773,392	59,924	2,302,583
2015-04	2015-10	20.60%	22.73%	2.12%	36.65%	2.28%	61.21%
		830,463	916,305	85,842	1,477,551	92,034	2,467,079
2015-10	2016-04	81.84%	84.16%	2.31%	91.03%	5.16%	6.64%
		4,013,131	4,126,835	113,704	4,464,119	253,261	325,678

Table 4. Training in two consecutive years and evaluating on a third. Training maximizing F1.

Train		Eval	System								
Source	Target	Target	Constant			Add-only			Del-only		
			Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
2010-3.6	2011-3.7	2012-3.8	47.03	19.02	**27.09**	56.82	52.34	**54.49**	78.04	41.17	**53.91**
		2013-3.9	70.10	18.96	**29.85**	73.98	44.43	**55.52**	83.99	51.38	**63.76**
		2014	43.05	22.94	**29.93**	31.80	38.00	**34.63**	49.38	75.49	**59.71**
		2015-04	29.11	23.38	**25.93**	13.79	33.55	**19.55**	55.93	67.60	**61.22**
		2015-10	35.06	19.05	**24.69**	18.60	30.36	**23.07**	48.66	70.30	**57.51**
		2016-04	84.16	11.48	**20.20**	84.53	35.48	**49.98**	93.47	59.10	**72.41**
2011-3.7	2012-3.8	2013-3.9	68.30	66.69	**67.49**	73.06	66.48	**69.61**	81.16	83.30	**82.22**
		2014	25.13	73.11	**37.41**	34.78	74.64	**47.46**	37.88	89.05	**53.15**
		2015-04	13.67	72.51	**23.00**	15.60	74.65	**25.81**	44.98	87.58	**59.43**
		2015-10	21.84	79.63	**34.29**	22.66	78.69	**35.19**	38.92	89.40	**54.23**
		2016-04	84.44	78.84	**81.54**	85.61	81.26	**83.38**	92.49	87.36	**89.85**
2012-3.8	2013-3.9	2014	28.60	91.60	**43.59**	37.29	94.06	**53.41**	38.57	94.45	**54.77**
		2015-04	15.60	90.91	**26.64**	16.92	94.89	**28.72**	45.50	90.62	**60.59**
		2015-10	23.49	92.60	**37.47**	24.06	95.53	**38.44**	39.49	91.95	**55.25**
		2016-04	84.53	85.54	**85.03**	84.93	92.32	**88.47**	92.75	88.58	**90.62**
2013-3.9	2014	2015-04	39.34	82.54	**53.28**	29.66	90.93	**44.73**	75.81	60.85	**67.51**
		2015-10	51.20	85.04	**63.92**	36.41	90.59	**51.95**	59.05	75.68	**66.33**
		2016-04	90.26	40.54	**55.95**	85.63	57.02	**68.45**	94.52	53.62	**68.42**
2014	2015-04	2015-10	63.18	57.75	**60.34**	63.20	78.61	**70.07**	71.66	63.78	**67.49**
		2016-04	87.67	20.31	**32.98**	91.76	33.37	**48.94**	92.73	70.61	**80.17**
2015-04	2015-10	2016-04	90.22	39.27	**54.72**	91.86	42.06	**57.70**	94.12	41.50	**57.60**

Table 5. First type analysis: for all types appearing in all years, pick the top most frequent 50 and see their overall precision / recall / F1 over all three consecutive years. Rows in italic are 20% above average in F1, rows in bold are 20% below average.

Type	Count	Constant			Add-only			Del-only		
		Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
AVERAGE	482,492	60.91	65.37	63.06	57.24	53.54	55.33	64.01	64.72	64.36
owl Thing	5,019,008	65.02	66.48	65.74	56.83	51.09	53.81	66.51	68.78	67.63
Person	3,412,161	**37.82**	**47.54**	**42.13**	**31.30**	**28.01**	**29.56**	**51.96**	**48.15**	**49.98**
Settlement	1,686,605	49.07	55.54	52.11	**40.99**	**31.44**	**35.59**	**46.83**	**35.48**	**40.37**
Place	1,682,897	53.76	80.45	64.45	46.04	51.83	48.77	49.18	57.33	52.95
Village	727,057	57.55	48.11	52.41	**43.80**	**25.18**	**31.98**	**52.13**	**30.08**	**38.15**
PersonFunction	648,343	*88.44*	*88.56*	*88.50*	*87.90*	*79.30*	*83.38*	*88.76*	*98.70*	*93.47*
SportsPerson	639,620	74.73	74.36	74.55	*72.58*	*72.25*	*72.41*	*91.85*	*96.26*	*94.00*
Athlete	608,903	50.60	57.10	53.65	**29.11**	**35.54**	**32.00**	**38.97**	**57.46**	**46.45**
Species	599,425	*75.80*	*75.56*	*75.68*	*73.25*	*74.93*	*74.08*	73.34	75.46	74.38
Eukaryote	579,287	75.34	75.17	75.25	*72.73*	*74.51*	*73.61*	72.76	75.02	73.87
Album	578,843	*79.85*	*75.83*	*77.79*	*75.15*	*72.84*	*73.98*	*77.59*	*88.28*	*82.59*
Organisation	547,605	51.14	60.83	55.56	46.78	52.02	49.26	52.50	65.41	58.25
Insect	514,096	*78.97*	*84.93*	*81.84*	*75.92*	*85.19*	*80.29*	*79.04*	*82.70*	*80.83*
SoccerPlayer	501,995	**31.63**	**19.51**	**24.13**	**10.22**	**2.37**	**3.85**	**25.63**	**11.56**	**15.94**
Film	439,197	*74.07*	*80.18*	*77.01*	61.83	65.88	63.79	68.29	77.36	72.54
Animal	435,377	*75.47*	*78.81*	*77.10*	*72.81*	*78.14*	*75.38*	72.69	79.47	75.93
MusicalWork	389,151	69.17	53.34	60.23	62.77	49.98	55.65	64.77	80.89	71.93
ArchitectStruc	318,087	54.22	79.84	64.58	45.08	55.16	49.61	49.18	65.08	56.03
Plant	272,468	76.63	72.96	74.75	*73.69*	*71.10*	*72.37*	75.35	68.92	71.99
Building	269,209	63.36	68.37	65.77	52.00	44.95	48.22	55.12	52.11	53.57
Company	252,438	73.66	77.13	75.36	*69.18*	*72.39*	*70.75*	73.88	79.03	76.37
MusicalArtist	234,646	**27.91**	**28.66**	**28.28**	**28.20**	**24.08**	**25.98**	**37.01**	**23.01**	**28.38**
Town	232,155	**44.94**	**32.61**	**37.80**	**34.67**	**17.86**	**23.58**	**43.37**	**23.75**	**30.69**
Artist	228,849	44.47	62.81	52.07	**33.96**	**46.72**	**39.33**	**42.72**	**55.96**	**48.45**
OfficeHolder	224,708	**31.13**	**25.94**	**28.29**	**16.50**	**12.18**	**14.01**	**22.51**	**13.22**	**16.65**
Single	216,311	61.07	53.82	57.22	54.06	42.39	47.52	53.20	57.94	55.47
Band	163,539	**30.43**	**25.33**	**27.65**	**24.89**	**16.62**	**19.93**	**36.54**	**29.68**	**32.76**
Book	160,711	**40.86**	**32.28**	**36.07**	**38.77**	**31.03**	**34.47**	**50.24**	**50.36**	**50.30**
Mollusca	155,990	75.24	71.29	73.21	*73.73*	*73.05*	*73.38*	*75.60*	*79.30*	*77.40*
School	143,682	**39.85**	**33.81**	**36.58**	**35.18**	**26.34**	**30.13**	**41.13**	**38.76**	**39.91**
TelevisionShow	142,129	63.72	64.11	63.92	55.28	50.22	52.63	64.36	69.32	66.75
MilitaryPerson	140,326	**22.63**	**14.99**	**18.04**	**15.11**	**7.31**	**9.85**	**30.50**	**20.88**	**24.78**
Ship	123,098	**46.53**	**30.97**	**37.19**	**30.95**	**9.40**	**14.42**	**27.37**	**28.81**	**28.07**
Station	121,190	**49.14**	**49.32**	**49.23**	**33.93**	**22.69**	**27.19**	**38.78**	**35.19**	**36.90**
River	120,146	74.50	70.20	72.29	*71.90*	*62.40*	*66.81*	*83.51*	*81.45*	*82.47*
Politician	119,184	**40.72**	**45.77**	**43.10**	**33.65**	**33.89**	**33.77**	**40.36**	**35.40**	**37.71**
AdminRegion	118,995	*74.76*	*86.82*	*80.34*	66.75	58.77	62.51	73.99	60.68	66.68
Writer	118,306	**45.34**	**53.65**	**49.15**	**39.74**	**33.31**	**36.24**	**47.71**	**33.35**	**39.26**
WrittenWork	116,619	**46.79**	**54.21**	**50.23**	43.00	48.75	45.69	53.27	71.48	61.04
EducationalInst	116,006	46.49	61.64	53.00	42.19	51.94	46.57	46.93	64.43	54.30
City	111,997	**32.94**	**21.21**	**25.80**	**23.74**	**10.80**	**14.84**	**36.70**	**16.97**	**23.21**

Table 6. Year drill-down for "bad" types (types 20% below average in F1) on the add-only category. The majority are related to either people (P) or location (L).

Type	2012-3.8	2013-3.9	2014	2015-04	2015-10	2016-04
(P) Artist	0.53	0.72	0.52	0.07	0.38	0.07
(P) Athlete	0.39	0.64	0.61	0.01	0.16	0.14
Band	0.66	0.49	0.00	0.36	0.27	0.01
(P) BaseballPlayer	0.00	0.63	0.00	0.17	0.17	0.01
Book	0.14	0.47	0.22	0.43	0.51	0.34
(L) City	0.46	0.45	0.16	0.55	0.16	0.10
(P) MilitaryPerson	0.13	0.31	0.40	0.07	0.22	0.04
(P) MusicalArtist	0.55	0.55	0.01	0.06	0.48	0.02
(P) OfficeHolder	0.36	0.36	0.47	0.07	0.05	0.05
(P) Person	0.46	0.66	0.62	0.13	0.04	0.10
(P) Politician	0.50	0.59	0.44	0.12	0.55	0.17
(L) Road	0.08	0.79	0.03	0.47	0.55	0.19
School	0.39	0.63	0.16	0.41	0.34	0.25
(P) Scientist	0.71	0.84	0.87	0.09	0.19	0.05
(L) Settlement	0.80	0.68	0.16	0.71	0.63	0.14
Ship	0.26	0.32	0.17	0.20	0.13	0.62
(P) SoccerPlayer	0.33	0.44	0.07	0.02	0.07	0.06
(L) Station	0.74	0.72	0.23	0.57	0.37	0.27
(L) Town	0.73	0.56	0.19	0.59	0.63	0.06
(L) Village	0.95	0.83	0.23	0.27	0.81	0.02
(P) Writer	0.62	0.83	0.78	0.12	0.30	0.08
WrittenWork	0.29	0.55	0.47	0.87	0.93	0.84

Table 7. Year drill-down for "good" types (types 20% above average in F1) on the add-only category.

Type	2012-3.8	2013-3.9	2014	2015-04	2015-10	2016-04
AdministrativeRegion	0.88	0.92	0.65	0.79	0.51	0.81
Album	0.40	0.28	0.86	0.79	0.86	0.91
Animal	0.59	0.92	0.69	0.89	0.90	0.93
AutomobileEngine	0.00	0.53	0.81	0.77	0.85	0.86
BodyOfWater	**0.75**	**0.82**	**0.72**	**0.75**	**0.75**	**0.84**
Company	0.32	0.38	0.81	0.79	0.81	0.87
Eukaryote	0.49	0.93	0.70	0.87	0.72	0.71
Film	0.65	0.59	0.81	0.72	0.80	0.84
Fish	0.59	0.93	0.68	0.80	0.87	0.72
Insect	**0.77**	**0.95**	**0.76**	**0.74**	**0.79**	**0.84**
Mollusca	0.04	0.86	0.71	0.76	0.77	0.81
PersonFunction	0.00	0.93	0.93	0.82	0.94	0.97
Plant	0.15	0.96	0.69	0.82	0.79	0.77
River	0.71	0.80	0.86	0.54	0.64	0.65
Species	0.49	0.93	0.70	0.88	0.88	0.91
SportsTeamMember		0.52	0.17	0.46	0.69	0.89

Table 8. Example predictions and mispredictions, using 2015-04 → 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→Perth
	added location→Australia

Incorrectly predicted as add-only	
Beverly_Hills_Handicap	disappears due to name change
First_Ward_Park	disappears due to name change
2012_Shonan_Bellmare_season	changes league→2012_J._League_Division_2
	to league→2012_J.League_Division_2

6 Application to Natural Language Generation

In Natural Language Generation [17], Referring Expressions Generation (REG), is the task of, given an entity (the **referent**) and a set of competing entities (the **set of distractors**), creation of a mention to the referent so that, 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 instances of REG algorithms by means of different versions of an ontology [8]. In [7] 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.

For that task, 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 in 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).

We plan to extend the current model with a specific model for additions and deletions using techniques from statistical machine translation [11] and investigate techniques based on knowledge embedding models [22]. In general, mining knowledge from changes in the data has been a successful meta-heuristic in Natural Language Generation [6].

7 Conclusions

Working on the problem of predicting changes in large knowledge graphs, we have found promise on a logistic regression approach to detect invariant nodes, particularly nodes within the graph that will not have any edges deleted or changed (add-only nodes). Our experiments employed eight different versions of DBpedia, a large scale knowledge graph with several million nodes. Our results show that for updates similar in size, models training in past data can help identify add-only nodes in future data.

From our experiments, we believe better data in terms of stable extraction scripts and equally spaced regular intervals is needed rather than relying in the data dumps graciously provided by the DBpedia project. Moreover, our experiments suggest that enriching the training features with type information could be a valuable direction. Alternatively, for future experiments, it might suffice to focus on people and locations as both types are plentiful and capture fine grained issues that should allow differentiating better models of the data.

We believe our techniques can be of interest to anomaly detection for security purposes and spam detection [20] and for maintenance of the data. The add-only nodes and relations can be pre-cached using more efficient data structures such as perfect hashes [3]. For example, dividing storage and algorithms for entities of different freshness is a standard solution the recommendation systems world [16] (Chapter 5: "Taking recommenders to production").

Acknowledgments

The authors would like to thank the Secretaria de Ciencia y Tecnica of Cordoba Province for support and Annie Ying and the anonymous reviewers for helpful comments and suggestions. They would also like to extend their gratitude to the organizers of the SIMBig Symposium.

References

1. Antoniou, G., Van Harmelen, F.: Web ontology language: Owl. In: Handbook on ontologies, pp. 67–92. Springer (2004)
2. Armbrust, M., Xin, R.S., Lian, C., Huai, Y., Liu, D., Bradley, J.K., Meng, X., Kaftan, T., Franklin, M.J., Ghodsi, A., et al.: Spark sql: Relational data processing in spark. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. pp. 1383–1394. ACM (2015)
3. Botelho, F.C., Ziviani, N.: External perfect hashing for very large key sets. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management. pp. 653–662. CIKM '07, ACM, New York, NY, USA (2007), http://doi.acm.org/10.1145/1321440.1321532
4. Cheng, S., Termehchy, A., Hristidis, V.: Efficient prediction of difficult keyword queries over databases. IEEE Transactions on Knowledge and Data Engineering 26(6), 1507–1520 (June 2014)

5. Drury, B., Valverde-Rebaza, J.C., de Andrade Lopes, A.: Causation generalization through the identification of equivalent nodes in causal sparse graphs constructed from text using node similarity strategies. In: Proc. of SIMBig, Peru (2015)
6. Duboue, P.A., McKeown, K.R.: Statistical acquisition of content selection rules for natural la nguage generation. In: Proceedings of the 2003 Conference on Empirical Methods for Natural Language Processing EMNLP-2003. Sapporo, Japan (Jul 2003)
7. Duboue, P.A., Domínguez, M.A.: Using Robustness to Learn to Order Semantic Properties in Referring Expression Generation, pp. 163–174. Springer International Publishing, Cham (2016)
8. Duboue, P.A., Domınguez, M.A., Estrella, P.: On the robustness of standalone referring expression generation algorithms using rdf data. WebNLG 2016 p. 17 (2016)
9. Eder, J., Koncilia, C.: C.: Modelling changes in ontologies. In: In: Proceedings of On The Move - Federated Conferences, OTM 2004, Springer (2004) LNCS 3292. pp. 662–673 (2004)
10. Kauppinen, T., Hyvnen, E.: Modeling and reasoning about changes in ontology time series. In: Integrated Series in Information Systems. pp. 319–338. Springer-Verlag (2007)
11. Koehn, P.: Statistical Machine Translation. Cambridge University Press, New York, NY, USA, 1st edn. (2010)
12. Lassila, O., Swick, R.R., Wide, W., Consortium, W.: Resource description framework (rdf) model and syntax specification (1998)
13. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal 6(2), 167–195 (2015)
14. Li, X., Zhou, W.: Performance comparison of hive, impala and spark sql. In: Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2015 7th International Conference on. vol. 1, pp. 418–423. IEEE (2015)
15. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al.: Mllib: Machine learning in apache spark. The Journal of Machine Learning Research 17(1), 1235–1241 (2016)
16. Owen, S., Owen, S.: Mahout in action. Manning Shelter Island, NY (2012)
17. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press (2000)
18. Rula, A., Panziera, L., Palmonari, M., Maurino, A.: Capturing the currency of dbpedia 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. (2014)
19. Stefanović, D., McKinley, K.S., Moss, J.E.B.: Age-based garbage collection. ACM SIGPLAN Notices 34(10), 370–381 (1999)
20. Tsai, C.F., Hsu, Y.F., Lin, C.Y., Lin, W.Y.: Intrusion detection by machine learning: A review. Expert Systems with Applications 36(10), 11994–12000 (2009)
21. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM 57(10), 78–85 (2014)
22. Xie, Q., Ma, X., Dai, Z., Hovy, E.: 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 (2017)