Predicting Invariant Nodes

in

Large Scale Semantic Graphs

Damin Barsotti, Martín Domínguez , **Pablo Duboue**

Facultad de Matemática, Astronomía y Física
Universidad Nacional de Córdoba
Córdoba, Argentina

Our Work at a Glance (AI)

- Our final objetive is to predict changes in **Wikipedia**.

- To this goal, in this work we identify automatically those entities that remain constant in a certain time period.

- Also, we learn to identify those entities that change only by adding information.

- We have moderate success, obtaining above 90% precision with recall up to 58%.

Our Work at a Glance (NLP)

- We want to evaluate NLG algorithms under ontology changes.
  - In particular, algorithms that produce referring expressions for a given entity.
- As an ontology, we use DBpedia.
  - Given that the content in Wikipedia pages is stored in a structured way. It is a knowledge multi-graph derived from the Wikipedia.
  - Freely available to download in the form of *dumps*.
  - These dumps contain the information in a language called Resource Description Framework (RDF).
- We train two different classifiers to predict invariant and add-only nodes.
  - For two consecutive dumps, we train a classifier.
  - We evaluate prediction in the next dump.

Data: DBpedia

- DBpedia, an ontology curated from Wikipedia infoboxes
  - RDF expressions are known as triples, where the subject denotes the resource being described, the predicate denotes a characteristic of the subject.
  - A collection of such RDF declarations can be formally represented as a labeled directed multi-graph, naturally appropriate to represent ontologies.

Data Size

- DBpedia versions used.

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

- Add only and constant

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

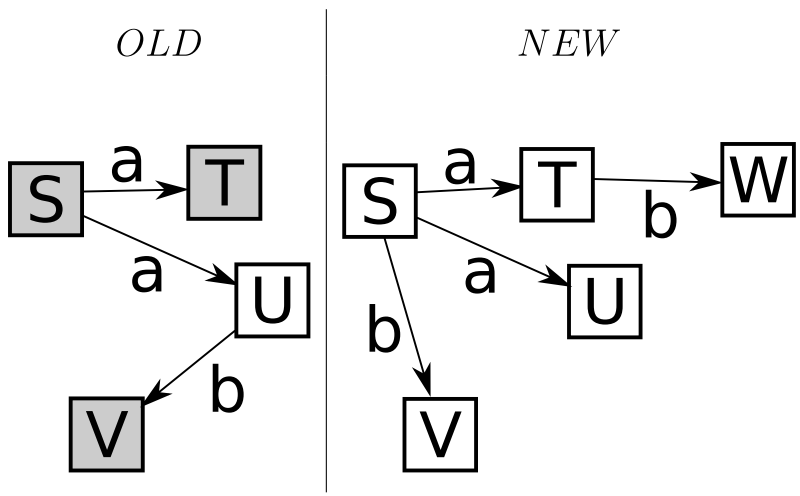
Methods

- Prediction System implemented on Apache Spark
  - Given two consecutive DBpedia Dumps, OLD and NEW.
  - We build a Feature vector for a classification problem.
  - The feature vector itself contains binary features indicating whether or not a given relation-object holds for the subject in OLD.
    - Each relation-object pair generates a feature.
  - add-only: $\{(V_i, O_i)\}_{OLD} \subseteq \{(V_i, O_i)\}_{NEW}$
  - constant: $\{(V_i, O_i)\}_{OLD} = \{(V_i, O_i)\}_{NEW}$

- Represent each DBpedia Dump Spark SQL Datasets.
  - Extract the appropriate views and train a logistic regression classifier using MLib.

Example

- DBpedia Dumps



- Feature vector example

Node	Features	Target	
S	a=T, a=U	add-only	$\neg$ constant
T	$\emptyset$	add-only	$\neg$ constant
U	b=V	$\neg$ add-only	$\neg$ constant
V	$\emptyset$	add-only	constant

Results

- Experimental setup
  - Using tree consecutive dumps, $G_{y_1}, G_{y_2}, G_{y_3}$,
  - built a model M on $G_{y_1} \to G_{y_2}$
  - apply M on G_{y_2}, obtaining G'_{y_3}
  - evaluation: compare G'_{y_3} and G_{y_3}.

- Experimental results
  - As our numbers were obtained by optimizing F1 on a binary classification problem, precision and recall are dual.

Results

Train		Eval	System			
Source	Target	Target	Add-only		Constant	
			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

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

Example Results

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 in 2016 due to name change to Red_Carpet_Handicap)
First_Ward_Park	disappears due to name change to First_Ward_Park_(Charlotte,_North_Carolina)
2012_Shonan_Bellmare_season	changes league→2012_J._League_Division_2 to league→2012_J.League_Division_2

Conclusion and Future Works

- REG: In Natural Language Generation, Referring Expressions Generation (REG), is the task that, given an entity (the **referent**) and a set of competing entities (the **set of distractors**), create a mention to the referent such that, in the eyes of the reader, it is clearly distinguishable from any other entity in the 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
  - 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 [1].
  - In [1] 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.

Conclusion and Future Works

- How useful are these results for predicting changes?
  - For the task "Predict Wikipedia Changes", predicting add-only and constant nodes help us immediately with the performance of the prediction 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).
  - Our high precision results will then carry over to direct improvements on our full system.

References

[1] Pablo Ariel Duboue and Martin Ariel Domínguez. *Using Robustness to Learn to Order Semantic Properties in Referring Expression Generation*, pages 163–174. Springer International Publishing, Cham, 2016.