# Using Embeddings to Predict Changes in Large Semantic Graphs

Damián Barsotti and Martín Ariel Domínguez

Group of Analysis and Processing of Large Social and Semantic Networks
FaMAF, Universidad Nacional de Córdoba, Argentina
{damian, mdoming}@famaf.unc.edu.ar

**Abstract.** Understanding and predicting how large knowledge graphs change over time is as difficult as it is useful. An important subtask to address this artificial intelligence challenge is to characterize and predict three types of nodes: add-only nodes that can solely add up new edges, constant nodes whose edges remain unchanged, and del-only nodes whose edges can only be deleted. In this work, we improve previous prediction approaches by using word embeddings from NLP to identify the nodes of the large semantic graph and build a Logistic Regression model. We tested the proposed model in different versions of DBpedia and obtained the following prediction improvements on F1 measure: up to 10% for add-only nodes, close to 15% for constant nodes, and close to 22% for del-only nodes.

**Keywords:** Semantic Graphs, Big Data, Graph Embeddings, Machine Learning

## 1 Introduction

We are interested in understanding and predicting how large knowledge graphs change over time to address various technological challenges, such as infrastructure needs related to data growth. An important subproblem is predicting which nodes within the graph will not have any edges deleted or changed (add-only nodes) or undergo any changes at all (constant nodes) and which ones will not have any edges added or modified (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. For example, we see parallelisms between our work and the track of changes in other large graphs, in particular the object graphs in garbage collection systems. State of the art garbage collection singles out objects that survive multiple garbage collections [12] and stops considering them for deletion. It is this type of optimizations that we expect detection of invariable nodes will help in the semantic graphs updates.

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

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.

In previous work [1], we show a logistic regression approach that, using binary attribute-values as features, achieves 90%+ precision on DBpedia[1] changes, we implement this model using Apache Spark. The present work improves those results by using a technique to represent a node in a large semantic ontology. This approach uses the **word2vec** word embeddings model [13] to represent the nodes in a large multi-graph as features to feed a Machine Learning (ML) model. The models obtained improve, in most cases, previous results considerably as we will see in the next sections.

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. Our methods and results follow, closing with a conclusion, and possible research lines to continue.

## 2 Related Work

Mining graphs for nodes with special properties is not new to Big Data mining [4]. As DBpedia grows, much research is being conducted to exploiting this resource in AI-related tasks, and to model its changes. For example, there is research on modeling DBpedia's currency [14], 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 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 data 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 variations 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 [6] 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 [2].

[1] `http://dbpedia.org`

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 [8]. In their research, 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.

As far as we know, there are no approaches which predict the three types of nodes we use in our work, and test models using the complete DBpedia. For example, in [12], the authors only predict if an entity will change in the future. They obtained 66% and 75% of precision and recall, respectively, but the test set is a small subset of the DBPedia. In previous work [1] we developed models to predict add-only, constant, and del-only nodes. In that previous work, we obtain a logistic regression model, using binary characteristics to represent the nodes of interest, by the set of outgoing edge labels and the destination node. This work continues that line, with a different approach for the characterization of the nodes. To do so, the learned model represents nodes with vectors which correspond to the word embedding of labels from outgoing edges and labels from destination nodes. With this change, the resulting model achieves an improvement of up to %22. The idea of this featurization was inspired in **node2vec** [7], an algorithmic framework for learning continuous feature representations for nodes in a graph. They define a flexible notion of the neighborhood of a node and design a biased random walk procedure. In our approach, we used this idea to generate an embedding model, which characterizes a node $s$ by the embeddings of the outgoing verb and object from edges $< s, v, o >$. Those embeddings are used as features in ML models for add-only, constant and del-only nodes, more in deep details are given in Section 5.

## 3 Data

We use DBpedia, a large scale naturally occurring knowledge graph with a rich update history. It is a knowledge graph derived from the Wikipedia collaborative encyclopedia that started in January 2001 and, at present, it contains 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. DBpedia contains 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 [10]. 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. To perform our experiments, we consider all the DBpedia dumps available from 2010 to 2016[2].

These dumps contain the information in a language called Resource Description Framework (RDF) [9]. The WWW Consortium (W3C) has developed the RDF to encode the knowledge present in web pages, so that it is understandable and exploitable by agents during any information search. RDF is based on the concept of making

[2] `https://wiki.dbpedia.org/develop/datasets`

statements about (web) resources using expressions in the subject-verb-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 relationship between the subject and the object.

A collection $s_i o_i v_i$ of such RDF declarations can be formally represented as a labeled directed multi-graph $G$, defined as set of labeled edges $< s_i, v_i, o_i >$, where $s_i$ and $o_i$ are nodes and $v_i$ a label. This is naturally appropriate to represent ontologies.

Table 1, in column "source", 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.

## 4 Methods

Our prediction system is implemented using Apache Spark[3], the Stellar-Random-Walk library[4], the **word2vec** implementation [13], and the distributed Logistic Regression package in MLlib [11].

This procedure consists of three phases: (1) Embedding model for node representation, (2) Building the prediction model, and (3) Prediction. The phases are described as follows:

- **Embedding model for node representation**: for this phase, we use the ideas from [7]. We build an embedding model to represent nodes based on **word2vec** embeddings [13], which associate each element in an RDF triple with a $n$-dimensional vector, for a given $n$. For NLP models, the training material to generate the embedding model is a set of NLP "sentences". Analogously, in our model, we generate "utterances" by random walks in the semantic graph. Formally, given a multi-graph $G$ as described in the previous Section, for every subject $s$ such that $< s, v, o > \in G$ , we simulate a fixed number $n$ of random walks up to length $l$. Once we generate the random walks for all subjects $s$ in the multigraph $G$, we get a collection of "sentences" $M_G$ such that :
  "$s_0\ v_0\ o_0 \ldots s_k\ v_k\ o_k$" $\in M_G \iff < s_i, v_i, o_i > k \in G$ where: $s = s_0$, $o_i = s_{i+1}$, for $1 \leq i \leq$ and $k \leq l$[5].
  Once, the collection $M_G$ is completed, we run the **word2vec** training procedure, with a number $N$ as input parameter, which returns an embedding model $E_G$ that maps any $s$ (subject), $v$ (verb) or $o$ (objet) with a vector of dimension $N$.
- **Building the prediction model**: In this phase, we generate three different Logistic Regression models: add-only $LG_{add}$, constant $LG_{const}$, and del-only $LG_{del}$ nodes. As we describe in the previous Section, a list of dumps of DBpedia is available, each of them for a given date. Each dump defines a multi-graph $G_i$ represented as a set of labeled edges $< s_i, v_i, o_i >$. We use two consecutive dumps as training material,

[3] `https://spark.apache.org/`

[4] `https://github.com/data61/stellar-random-walk`

[5] Note that $k \leq l$ because $o_k$ could be a literal, date for example, then it has no outgoing edges.

and the third for evaluation. Let $G_i$ and $G_{i+1}$ be two consecutive dumps of DBpedia that we use as the training material. As any ML problem, we build a feature and target vectors. The feature vectors are the same for building $LG_{add}$, $LG_{const}$ and $LG_{del}$. We first build an embedding as in the previous phase, let $E_{G_i \cup G_{i+1}}$ be the embedding for the union of graphs $G_i \cup G_{i+1}$. Then, for every subject $s \in G_i$, let $v_1, o_1 \dots v_r, o_r$ be the verb and objects from the outgoing edges of node $s$. Then, we compute the maximum, average, and minimum of $v_i$, and also of $o_i$. In this way, using the average of vectors, the impact of the instability of **word2vec** models for unfrequent words is reduced. Also, during the **word2vec** training phase, terms with less than five occurrences are deleted. Finally, we concatenate those embedding vectors. Formally, that is:
$Av(E_{G_i \cup G_{i+1}}(v_1), \dots, E_{G_i \cup G_{i+1}}(v_r)) \bullet Av(E_{G_i \cup G_{i+1}}(o_1), \dots, E_{G_i \cup G_{i+1}}(o_r)) \bullet$
$Ma(E_{G_i \cup G_{i+1}}(v_1), \dots, E_{G_i \cup G_{i+1}}(v_r)) \bullet Ma(E_{G_i \cup G_{i+1}}(o_1), \dots, E_{G_i \cup G_{i+1}}(o_r)) \bullet$
$Mi(E_{G_i \cup G_{i+1}}(v_1), \dots, E_{G_i \cup G_{i+1}}(v_r)) \bullet Mi(E_{G_i \cup G_{i+1}}(o_1), \dots, E_{G_i \cup G_{i+1}}(o_r))$
Where $Av()$, $Ma()$ and $Mi()$ calculate the average, maximum and minimum between vectors, and $\bullet$ is the concatenation. The resulting feature vector has $6 * N$ dimensions, where $N$ is the selected dimension for **word2vec** embeddings.
To obtain the binary target vector for a given node with subject $s$:
add-only: $\{(s, v_i, o_i) : (s, v_i, o_i) \in G_i\} \subseteq \{(s, v_i, o_i) : (s, v_i, o_i) \in G_{i+1}\}$
constant: $\{(s, v_i, o_i) : (s, v_i, o_i) \in G_i\} = \{(s, v_i, o_i) : (s, v_i, o_i) \in G_{i+1}\}$
del-only: $\{(s, v_i, o_i) : (s, v_i, o_i) \in G_{i+1}\} \subseteq \{(s, v_i, o_i) : (s, v_i, o_i) \in G_i\}$.
We feed the Mlib with these features and target vectors, and compute the three models $LG_{add}$, $LG_{const}$ and $LG_{del}$.

– **Prediction**: Finally, in this phase we can predict add-only, constant, and del-only nodes in the unseen data from the next dump $G_{i+2}$.

**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 del-only.

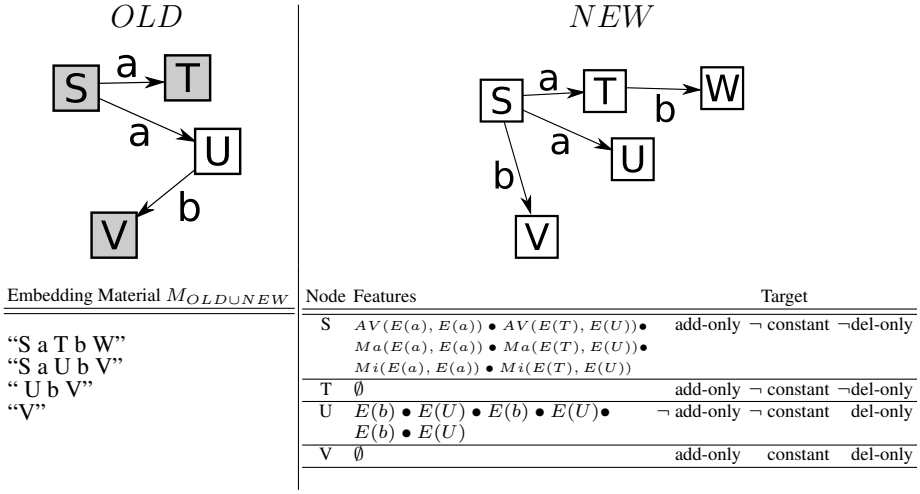


Fig. 1 shows a small example of feature and class extraction. A four-node graph $G_i$ that we call $OLD$ evolves into a five node graph $G_{i+1}$ ($NEW$). The target for each

node is computed over $OLD$, using three binary features. The features are calculated from the resulting embedding model $E_{OLD \cup NEW}$(renamed as $E$) computed from $M_{OLD \cup NEW}$ training material.

**Table 1.** Training in two consecutive years and evaluating on a third. Training maximizing F1.

| | Train | | Eval | System | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | Source | | Target | Add-only | | | Constant | | | Del-only | | |
| | $G_{y1}$ | $G_{y2}$ | $G_{y3}$ | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ |
| res-2018 | 2010-3.6 | 2011-3.7 | 2012-3.8 | .57 | .52 | .55 | .47 | .19 | .27 | .78 | .41 | .54 |
| d64 | | | | .60 | .51 | .55 | .51 | .23 | .32 | .81 | .43 | **.56** |
| d128 | | | | .62 | .52 | .56 | .54 | .25 | **.34** | .84 | .39 | .54 |
| d256 | | | | .61 | .53 | **.57** | .55 | .24 | .33 | .84 | .40 | .54 |
| res-2018 | 2011-3.7 | 2012-3.8 | 2013-3.9 | .73 | .67 | .70 | .68 | .67 | .68 | .81 | .83 | .82 |
| d64 | | | | .70 | .68 | .69 | .67 | .71 | .69 | .81 | .90 | **.85** |
| d128 | | | | .71 | .70 | .70 | .71 | .69 | **.70** | .81 | .89 | **.85** |
| d256 | | | | .77 | .68 | **.73** | .75 | .66 | **.70** | .80 | .77 | .79 |
| res-2018 | 2012-3.8 | 2013-3.9 | 2014 | .37 | .94 | .53 | .29 | .92 | .44 | .39 | .95 | .55 |
| d64 | | | | .36 | .98 | .53 | .33 | .97 | **.49** | .40 | .98 | .57 |
| d128 | | | | .38 | .97 | **.55** | .32 | .96 | **.49** | .41 | .97 | **.58** |
| d256 | | | | .38 | .98 | **.55** | .29 | .96 | .45 | .41 | .97 | **.58** |
| res-2018 | 2013-3.9 | 2014 | 2015-04 | .30 | .91 | .45 | .39 | .83 | .53 | .76 | .61 | .68 |
| d64 | | | | .33 | .94 | .48 | .48 | .87 | .62 | .83 | .63 | .72 |
| d128 | | | | .33 | .95 | **.49** | .49 | .89 | **.63** | .85 | .63 | .72 |
| d256 | | | | .32 | .96 | .48 | .47 | .88 | .61 | .86 | .63 | **.73** |
| res-2018 | 2014 | 2015-04 | 2015-10 | .63 | .79 | .70 | .63 | .58 | .60 | .72 | .64 | .68 |
| d64 | | | | .76 | .78 | .77 | .75 | .73 | .74 | .70 | .82 | .75 |
| d128 | | | | .73 | .86 | .79 | .74 | .77 | .75 | .73 | .81 | .77 |
| d256 | | | | .74 | .86 | **.80** | .75 | .76 | **.75** | .78 | .79 | **.80** |
| res-2018 | 2015-04 | 2015-10 | 2016-04 | .92 | .42 | **.58** | .90 | .39 | **.55** | .94 | .42 | .58 |
| d64 | | | | .94 | .35 | .51 | .92 | .31 | .47 | .94 | .65 | .77 |
| d128 | | | | .93 | .38 | .54 | .92 | .34 | .49 | .94 | .67 | .78 |
| d256 | | | | .93 | .39 | .55 | .92 | .35 | .50 | .95 | .68 | **.79** |

## 5 Results

Using the machine learning method described in the previous section we take three consecutive years, $G_{y_1}, G_{y_2}, G_{y_3}$, build a model $M$ on $G_{y_1} \cup G_{y_2}$, apply $M$ on $G_{y_2}$, obtaining $G'_{y_3}$ and evaluate it by comparing it to $G_{y_3}$. For features generation, we use random walks with a maximum of 10 walks per node, and a maximum depth of 80. We generate prediction models $M$ for 64, 128, 256 dimensions of embedding models. Table 1 shows the results compared with the baseline results, labeled as "res-2018", obtained in previous work [5]. The comparison is performed with the new prediction models calculated by using embedding vectors of dimensions 64, 128, and 256. We can see that for del-only nodes the improvement varies between 2% to 20%, depending on the years predicted. For add-only and constant nodes, the increase ranges from 1% to 10%. For del-only nodes, we achieve the best improvements, even for the last dump (2016), where the other models decrease their performance about 5%.

From the table, we can also see that detecting constant nodes, on the other hand, is a much more difficult task that might be ill-defined given the nature of the updates discussed in the introduction. Finally, in Table 2, we show some examples of right prediction and mispredictions for add-only nodes.

**Table 2.** Example of predictions and mispredictions, using 2015-04 → 2015-10 as training and tested on 2016-04 with model 'd128'.

| | |
|---|---|
| **Correctly predicted add-only** | |
| 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*** |
| Basie_Land changes | ***changes*** previousWork → On_My_Way_and_Shoutin'_Again! |
| | ***to*** previousWork→Ella_and_Basie! |
| **Correctly predicted as not add-only node.** | |
| 2012_Shonan_Bellmare_season | ***changes*** league→2012_J._League_Division_2 |
| | ***to*** to league→2012_J.League_Division_2 |

## 6 Conclusions and Future Work

In this work, we improve the performance of the results obtained in previous work [1] for predicting node changing in different temporal dumps of DBpedia. For this purpose, we use a novel way to characterize the nodes of a large semantic graph using NLP concepts: word embedding. With this way of mining the graph, in most cases, we obtain significant improvements, up to 22%, while we get a small drop in performance for add-only and constant nodes for the last available dump in the conventional DBpedia. We think that this model has less performance than all previous ones, due to the training material have less variability. The reason is that the DBpedia dumps used to obtain the classifier belongs to the same year, and for all previous cases it used different years.

There are several possible lines of work to continue this publication. In this sense, to further improve the models, we plan to conduct research about the convolution function for node embeddings. We have used a simple average of the vectors of the component words, but there are other more sophisticated functions, such as the weighted average by the inverse document frequency (IDF) [15].

Another possibility of improving the results is to use an end-to-end graph-based neural network model, including Recurrent Neural Networks for the node representation. This approach solves an NLP task, the relation extraction in a text, see [3].

Besides, we are interested in testing DBpedia dumps after 2016, which are available in a new version called DBpedia Live. This new version makes it possible to obtain dumps on the desired date and offers pre-calculated dumps after 2016 [6].

On the other hand, we plan to use these results in other lines of research. In [5] we presented experiments on two types of entities (people and organizations) and, using

[6] `https://databus.dbpedia.org/dbpedia/mappings`

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.

In [5], we explored the robustness for the particular case of Referring Expressions Generation (REG) algorithms through different versions of an ontology. Our current work is part of a plan to simulate natural perturbations on the data to find the conditions in which REG algorithms start to fail (for example, a simulated DBpedia of 25 years in the future). For general changes prediction of DBpedia, we started using the ideas of "Disentangled" representation as in [16], we split large scale graph evolution in simple problems, add-only, constant and del-only nodes prediction.

## References


1. Barsotti, D., Dominguez, M.A., Duboue, P.: Predicting invariant nodes in large scale semantic knowledge graphs. In: SIMBig 2017. CCIS series. Springer (2018)
2. 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 (2014)
3. Christopoulou, F., Miwa, M., Ananiadou, S.: A walk-based model on entity graphs for relation extraction. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). ACL, Melbourne, Australia (2018)
4. 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: Proceedings of SIMBig'15. pp. 58–65 (2015)
5. Duboue, P.A., Domínguez, M.A.: Using Robustness to Learn to Order Semantic Properties in Referring Expression Generation, pp. 163–174. Springer International (2016)
6. Eder, J., Koncilia, C.: C.: Modelling changes in ontologies. In: In: Proceedings of On The Move - Federated Conferences, OTM 2004, Springer LNCS 3292. pp. 662–673 (2004)
7. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. CoRR (2016)
8. 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)
9. Lassila, O., Swick, R.R., Wide, W., Consortium, W.: Resource description framework (rdf) model and syntax specification (1998)
10. 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)
11. 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)
12. Meroo-Peuela, A., Guret, C., Schlobach, S.: Release early, release often: Predicting change in versioned knowledge organization systems on the web. CoRR **abs/1505.03101** (2015)
13. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proc. of NIPS13 (2013)
14. 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) (2014)
15. Sanjeev Arora, Yingyu Liang, T.M.: A simple but tough-to-beat baseline for sentence embeddings. In: Proceeding of International Conference on Learning Representations, ICLR 2017, April 24 - 26, Toulon, France (2017)
16. Whitney, W.: Disentangled representations in neural models. CoRR **abs/1602.02383** (2016)