Information-theoretic Analysis of Entity Dynamics on the Linked Open Data Cloud

Chifumi Nishioka[1,2] and Ansgar Scherp[1,2]*

[1] Kiel University,
Christian-Albrechts-Platz 4, 24118 Kiel, Germany
[2] ZBW – Leibniz Information Centre for Economics,
Düsternbrooker Weg 120, 24105 Kiel, Germany
`{chni,asc}@informatik.uni-kiel.de`

Abstract. The Linked Open Data (LOD) cloud is expanding continuously. Entities appear, change, and disappear over time. However, relatively little is known about the dynamics of the entities, i. e., the characteristics of their temporal evolution. In this paper, we employ clustering techniques over the dynamics of entities to determine common temporal patterns. We define an entity as RDF resource together with its attached RDF types and properties. The quality of the clusterings is evaluated using entity features such as the entities' properties, RDF types, and pay-level domain. In addition, we investigate to what extend entities that share a feature value change together over time. As dataset, we use weekly LOD snapshots over a period of more than three years provided by the Dynamic Linked Data Observatory. Insights into the dynamics of entities on the LOD cloud has strong practical implications to any application requiring fresh caches of LOD. The range of applications is from determining crawling strategies for LOD, caching SPARQL queries, to programming against LOD, and recommending vocabularies for reusing LOD vocabularies.

1 Introduction

The Linked Open Data (LOD) cloud is a global information space to structurally represent and connect data. Since its advent in 2007, it has been continuously evolving and is covering a wide range of domains [19], today. Understanding the evolution of the LOD cloud is important for different applications such as LOD caching [22], indexing of distributed data sources [12], and query optimization [14]. For instance, Umbrich et al. [22] developed a query execution engine taking into account an analysis whether a dataset is static or dynamic, in order to automatically decide whether data should be retrieved from caches or from the original LOD cloud. The authors have compared two snapshots of a LOD dataset captured at two different points in time and analyzed which triples have been preserved, deleted, and added. Käfer et al. [10] quantified changes with respect to set of triples, set of links, and schema signature. Dividino et al. [5]

* Copyright held by the authors.

measured changes with respect to the schema information of a dataset. While the aforementioned works focused on evaluating changes between two different points in time, Dividino et al. [4] also analyzed the temporal dynamics of a dataset, i. e., the temporal evolution of the LOD cloud over the entire period. They presented detailed analyses of LOD evolution for thirteen LOD sources and represent the degree of evolution by a non-negative, single value. The result of the analyses enables to crawl LOD documents more efficiently [3].

In the realm of web search, knowledge bases contain information about different real-world objects or concepts commonly referred to as *entities*. Since the most popular type of queries contain entities [26], it is important to understand temporal dynamics of entities, in order to keep information of entities up to date and accurate. To best of our knowledge, few works have conducted an analysis of temporal dynamics in the entity level such as the early work of Ding et al. [2]. In terms of applications, triple stores are accessed by various web data applications via SPARQL queries. Martin et al. [13] showed that SPARQL query caching improve the performance of query engines. While they assumed that they could be aware of all updates of data, in practice it is not easy. In addition, recent profiling methods utilize entities from the LOD cloud. Schuhmacher et al. [20] proposed a document profiling method using DBpedia. Their method annotates a given document with DBpedia entities using a snapshot of DBpedia. It is important to take into account the temporal dynamics of entities, because DBpedia has been updated continuously and a part of information in a snapshot of DBpedia can quickly get stale.

In this paper, we analyze temporal dynamics of entities on the LOD cloud. To this end, we use snapshots of the LOD cloud over three years. We first introduce how to define entities on the LOD cloud. Subsequently, we measure the degree of changes of entities between two snapshots and represent the dynamics of entities as time series. We use two different representations of entities and two different distance measures to compute the degree of changes. In addition, we introduce different triple weighting methods, where each triple in the entity has different importance. We apply k-means++ clustering to find out temporal patterns of entity dynamics. We discover dominant temporal patterns in the clusters by using a periodicity detection algorithm by Elfeky et al. [6]. Subsequently, we investigate which features of entities control patterns of entity dynamics.

In Section 2, we review related work. Subsequently, we introduce basic formalization in Section 3. We compute and determine temporal patterns of entity dynamics and analyze the periodicities of them in Section 4. Section 5 describes the data. We provide the observed temporal patterns of entity dynamics and analyze the effects from entity features in Section 6, before concluding this work.

2 Related Work

In this section, we review works tackling with LOD dynamics. Before describing the related work, we distinguish "change" and "dynamics" along with Dividino et al. [4]. Changes of LOD sources are analyzed with regarding to their sets of triples, sets of entities, or schema signatures. For example, given two snapshots of a dataset captured at two different points in time, the change analysis at the triple level includes which triples from the previous snapshot have been preserved in the later snapshot, which triples have been deleted, or which ones have been added. On the other hand, the dynamics of a dataset involves a notion of how "fluid" a dataset is, i .e., how it behaves and evolves over a certain period of time. Thus, the dynamics involves the analysis of its development over more than two points in time.

Käfer et al. [10] provided the comprehensive analysis of the dynamics of LOD based on monitoring 86, 696 LOD documents for 29 weeks. They found out that 5.0% of documents had gone offline and 62.2% of documents had no change. In addition, they conducted the analysis on triple level. The result indicated that additions of triples are much more frequent than deletions. Furthermore, they observed that while object literals are the most dynamic element of triples, predicates (i .e., properties) and RDF types defined by the predicate `rdf:type` are static. They looked into the most dynamic predicates and identified that they were often about trivial time stamps. Dividino et al. [4] attempted to grasp the temporal dynamics of LOD sources. Beyond recent researches which focused on changes by comparing two snapshots of a source [10], Dividino et al. [4] analyzed dynamics of LOD sources. They proposed a monotone and positive function to represent the dynamics as a value. They conducted detailed analyses with respect to thirteen LOD sources and provided the statistics. Furthermore, Dividino et al. [3] evaluated strategies to keep LOD stored in a local repository up to date efficiently with limited bandwidth. Strategies define when to fetch data of which data source. The experiment revealed that strategies based on dynamics of data sources [4] performed best, compared to strategies based on the data source's age, PageRank, or size.

Below we describe works that conducted the analysis of entity dynamics. Umbrich et al. [23] formed a labeled directed graph based on LOD, where a node is an entity and an entity is represented by a subject URI. They analyzed entity dynamics using this graph. They applied k-means clustering to group entities with similar dynamics. After manual inspection, they observed that entities from same domains were often found in same clusters. However, they considered only whether there was a change or not and did not take into account the amount of changes of entities. Popitsch et al. [16] provided statistics about entity changes between two DBpedia snapshots with respect to four OWL classes (i .e., `person`, `organization`, `place`, `work`). In terms of OWL classes, entities belonging to the class `person` were active, because a lot of entities were removed and created. The number of entities belonging to the class `location` increased the most. The focus of the work by Popitsch [16] was not to analyze temporal dynamics of entities but to develop an effective entity change detection framework to avoid broken links. Thus, they have not conducted a more fine-grained analysis. Holzmann et al. [9] looked into entity evolution, but focused on changes of entity names in Wikipedia.

3 Basic Formalization

In this section, we briefly introduce definitions and notations. Table 1 summarizes the symbol notations and Table 2 shows a small example of LOD snapshots.

Data fetched from the LOD cloud is represented in the form of N-Triples[3]. A triple x of a dataset X is represented as $x = (s, p, o)$ where s, p, and o correspond to a subject, predicate, and object. Please note that we focus on analyzing triples and do not consider the contexts of the triples in this work, i. e., the sources where these triples where obtained. Furthermore, we define the sets of all possible URIs U, blank nodes B, and literals L. In a triple x, the subject $s \in U \cup B$ is a URI or a blank node, the predicate $p \in U$ a URI, and the object $o \in U \cup B \cup L$ a URI, a blank node or a literal. Functions *sub*, *pred*, and *obj* return the subject, predicate, and object of a given triple x, respectively. We assume that the data from the different LOD sources is captured at some point in time t. We define X_t as the set of triples (i .e., snapshot) captured at the point in time t and $\mathbb{X} = \{X_{t_1}, X_{t_2}, \dots, X_{t_n}\}$ as the collection of the snapshots at the different points in time. Using the example in Table 2, X_{t_1} contains three triples and each of X_{t_2} and X_{t_3} has five triples.

In order to investigate the temporal dynamics of the entities in the LOD cloud, we represent an entity as set of triples returned by a function ϵ which takes an entity key $e \in U \cup B$ and a point in time t as arguments. Below, we introduce the definitions of entities and the entity function. We distinguish two different representations of entities: $E^{O}_{e,t}$ denotes entities that are characterized by RDF types and outgoing properties and $E^{IO}_{e,t}$ additionally considers incoming properties.

Definition 1. An entity $E^{O}_{e,t}$ (Out) with RDF types and outgoing properties: *For a given entity key e and a point in time t, $E^{O}_{e,t} = \epsilon_O(e, t) = \{x \mid x \in X_t \wedge sub(x) = e\}$, where triples containing e as subject are grouped together as an entity.*

Definition 2. An entity $E^{IO}_{e,t}$ (InOut) with RDF types, outgoing properties and incoming properties: *For a given entity key e and a point in time t, $E^{IO}_{e,t} = \epsilon_O(e, t) = E^{O}_{e,t} \cup \{x \mid x \in X_t \wedge obj(x) = e\}$, where triples containing e as subject or object are grouped together as an entity.*

We derive entities $E_{e,t}$ for all unique URI $e \in U \cup B$ with respect to each snapshot X_t. We use z to describe the entity representation. In this work, $z = O$ or $z = IO$. In Table 2, there are three entities `db:Anne_Smith`, `db:John_Brown`, and `db:Green_Village` in X_{t_3} when $z = O$. On the other hand, there are four entities `db:Anne_Smith`, `db:John_Brown`, `db:Green_Village`, and `db:Green_University` in X_{t_3} when $z = IO$.

[3] `http://www.w3.org/TR/n-triples/`, last access on 11/05/2015

Table 1: Symbol Notation

t	a point in time
X_t	a set of triples captured at a point in time t
$\mathbb{X}$	the set of all X_t
$x = (s, p, o)$	a triple
s, p, o	a subject, a predicate, an object
sub, $pred$, obj	functions that returns s, p, and o for a given triple x
e	an entity key, where $e \in U$
$E_{e,t}$	an entity for a given entity key e at point in time t

Table 2: An example of snapshots.

X_{t_1}: a snapshot at time t_1		
db:Anne_Smith	db:location	db:Green_Village
db:Anne_Smith	db:works	db:Green_University
db:Green_Village	db:population	224123
X_{t_2}: a snapshot at time t_2		
db:Anne_Smith	db:location	db:Green_Village
db:Anne_Smith	db:works	db:Green_University
db:John_Brown	db:location	db:Green_Village
db:John_Brown	db:works	db:Green_Institute
db:Green_Village	db:population	223768
X_{t_3}: a snapshot at time t_3		
db:Anne_Smith	db:location	db:Green_Village
db:Anne_Smith	db:works	db:Green_University
db:John_Brown	db:location	db:Green_Village
db:John_Brown	db:works	db:Green_University
db:Green_Village	db:population	223540

4 Computing Temporal Patterns of Entity Dynamics

In this section, we describe how we determined the temporal patterns of entity dynamics. First, Section 4.1 describes how to weigh different triples in an entity. In Section 4.2, we explain how to measure a degree of changes of an entity between two successive snapshots and how to represent entity dynamics as time series. Section 4.3 provides the employed clustering method and optimization metric, in order to find out the most representative temporal patterns of entity dynamics. Subsequently, we investigate periodicities of the observed temporal patterns in Section 4.4.

4.1 Triple Weighting

We assume that triples of an entity have different importance [20]. For example, a triple (`Barack_Obama`, `dbp:vicepresident`, `Joe_Biden`) is considered more important than a triple (`Barack_Obama`, `rdf:type`, `foaf:Person`) for an entity `Barack_Obama`. Because there are a lot of entities whose `rdf:type` is `foaf:Person`, but a few entities which have a property `dbp:vicepresident`. Thus, we should give a larger weight on more important triples. For instance, it is not helpful to update a data cache due to a change of a trivial triple. Because in practical applications, only important facts of the entities encoded in a knowledge graph will be shown. Therefore, trivial changes like updating time-stamps have no effect on most applications. In fact, Käfer et al. [10] observed that most changes were updates of time-stamps, which are modeled as literals of a triple. Therefore, we give a weight $w(x)$ to each triple and analyze the entity dynamics considering these weights. Below, we introduce methods to weigh triples, starting from the baseline.

1. Baseline. We give the same weight to all triples.

$$w_{baseline}\left(x\right) = 1 \tag{1}$$

Schuhmacher et al. [20] developed several methods to weigh semantic relations (i .e., predicates) between entities for document modeling. Based on their work, we introduce a method to weigh triples. At the core of the weighting method lies the information-theoretic notion of information content (IC) [21]. Then, the IC of a specific variable $v = pred(x)$ can be computed as shown in Equation 2, where $P(v)$ is the probability that the random variable V shows the specific outcome v.

$$IC(v) = -\log(P(v)) \tag{2}$$

Based on IC, Combined Information Content (combIC) is proposed to weigh triples. Although the authors also introduced another method, Joint Information Content (jointIC), we only provide the results provided by combIC. The results of jointIC were very similar to those of combIC. We choose combIC, because it requires less computation and demonstrates better results [20].

2. Combined Information Content (combIC). Weights are computed as the sum of IC of a predicate and a object.

$$w_{combIC}\left(x\right) = IC\big(pred(x)\big) + IC\big(obj(x)\big) \tag{3}$$

Please note that the information theoretic weights are computed with respect to each snapshot, i .e., each point in time. Therefore, probabilities such as $P(pred(x))$ must not be 0 theoretically. Based on this, we can finally define an entity as a vector where each element is a weight of a triple.

Definition 3. Weighted entity vector $\vec{E}_{e,t}$
An entity for a given entity key e and a point in time t is represented as a vector described in Equation 4.

$$\vec{E}_{e,t} = \big(\mathbb{1}_e(x_{t,1}) \cdot w\left(x_{t,1}\right), \mathbb{1}_e(x_{t,2}) \cdot w\left(x_{t,2}\right), \ldots, \mathbb{1}_e(x_{t,1}) \cdot w\left(x_{t,|X_t|}\right)\big), \tag{4}$$

where $x_{t,i}$ denotes the i-th triple in all unique triples from X_t, and $\mathbb{1}_e(x)$ is an indicator function which returns 1 if $E_{e,t}$ contains a triple x and 0 if not.

Using the example shown in Table 2, $\vec{E}_{\texttt{db:Annee\_Smith},t_1} = (1,1,0)$, when using the baseline weighting method. Because $E_{\texttt{db:Annee\_Smith},t_1}$ contains the first and second triple of X_{t_1} and the indicator function $\mathbb{1}_{\texttt{db:Annee\_Smith}}$ returns 1 for the first and second triples and 0 for the last one.

4.2 Measuring Entity Dynamics

We first measure a degree of changes for a given entity key e and an entity representation z (i.e., Out or InOut shown in Section 3) between two successive snapshots, using a function δ. We introduce two variations of δ using cosine distance in Equation 5, and Euclidean distance in Equation 6.

$$\delta_{cosd}(e, z, t_1, t_2) = 1 - \frac{\vec{E^z_{e,t_1}} \cdot \vec{E^z_{e,t_2}}}{||\vec{E^z_{e,t_1}}|| \cdot ||\vec{E^z_{e,t_2}}||} \tag{5}$$

$$\delta_{euclidean}(e, z, t_1, t_2) = \sqrt{\sum_{x \in E^z_{e,t_1} \cup E^z_{e,t_2}} \big(w(x_{e,t_1}) - w(x_{e,t_2})\big)^2} \tag{6}$$

Equation 5 is based on cosine similarity that is widely used. We modify it to compute distance, since we focus on how much an entity is changed. If a triple is contained in E^z_{e,t_1} and not in E^z_{e,t_2}, the weight of that triple in E^z_{e,t_2} is 0 and vice versa. In Equation 6, $w(x_{e,t})$ denotes a weight of a triple x for an entity key e at a point in time t. Like Equation 5, if a triple is only contained in E^z_{e,t_1}, the weight of that triple at the point in time t_2 is 0 and vice versa. Finally, we represent the temporal dynamics of an entity as a series of entity changes as defined below.

Definition 4. Temporal dynamics of an entity $\Delta(e, z)$
The temporal dynamics of an entity are represented as a time series, where each element denotes the amount of changes between two successive snapshots.

$$\Delta(e, z) = (\delta(e, z, t_1, t_2), \delta(e, z, t_2, t_3), \ldots, \delta(e, z, t_{n-1}, t_n)) \tag{7}$$

We construct such an entity dynamics vector for all entities e. In order to find patterns of entity dynamics, we apply a clustering algorithm as described below.

4.3 Time-series Clustering

For clustering, we leave out entity dynamics vectors where all elements are equal 0 (i.e., time series with no changes over the entire observed period). As clustering method, we employ k-means++ [1] and use Euclidean distance as a distance measure, as our previous work [15]. Compared to the traditional k-means, k-means++ introduces an improved initial seeding [1]. In terms of the distance measure, an extensive evaluation conducted by Wang et al. [24] demonstrates that Euclidean distance is the most efficient measure for time series with a reasonably high accuracy.

We find the optimal number of clusters k between 2 and 10 using Average Silhouette [18]. Due to the computation cost, we randomly pick 0.1% of elements from each cluster and compute Average Silhouette over them. A higher value indicates a better clustering. We consider centroids of generated clusters as representative temporal patterns of entity dynamics.

4.4 Periodicity Detection

As results of clustering described in Section 4.3, we obtain time series that show representative temporal patterns of entity dynamics. We further analyze these time series by periodicity detection. Periodicity detection is the task to discover the pattern at which a time series is periodic. For instance, temporal sequences $(1, 3, 2, 1, 3, 2)$ and $(1, 2, 1, 2, 1, 2)$ have the periodicity of 3 and 2, respectively. Computing periodicity over the observed entity dynamics ensures generalizability of the observed temporal patterns. We employ a convolution-based algorithm proposed by Elfeky et al. [6]. The algorithm outputs periodicity candidates with confidence scores.

5 Dataset

We use the Dynamic Linked Data Observatory (DyLDO) dataset[4]. The DyLDO dataset has been created to monitor a fixed set of LOD documents on a weekly basis. The dataset is composed of 165 weekly snapshots over a period of three years from May 2012 to July 2015. The DyLDO dataset contains various well known and large LOD sources (e. g., `dbpedia.com`, `bbc.co.uk`) as well as less commonly known ones (e. g., `pokemon.com`). For more detailed information about the dataset, we refer to [11].

The dataset contains $27,788,902$ unique entities ($z = \mathit{Out}$) and $2,909,700$ unique entities ($z = \mathit{InOut}$) over the 165 snapshots. Figure 1 shows the distributions of entity frequencies for Out and InOut, respectively. According to Figure 1, almost 75% of the entities appear only in one snapshot in both entity representations Out and InOut. To conduct the time series clustering properly, we focus on $2,521,617$ entities in Out and $2,950,533$ entities in InOut that appear at more than 70% of snapshots.

6 Results and Analyses

We first report the observed temporal patterns of entity dynamics. Subsequently, we investigate entity features that define temporal patterns of entity dynamics.

6.1 Temporal Patterns of Entity Dynamics

Table 3 shows the resulting clusters for each condition. Since we use two different entity representations, two different distance measures, and two different triple weighting methods, we have in total eight conditions to compute time series and clustering. In terms of the optimal number of clusters k, generally Average Silhouette suggests that a lower value of k gives better clustering, as observed by Yang et al. [25]. Since we first remove all static entities (i.e., entities with no change) before clustering, the optimal number of clusters shown in Table 3 is

[4] `http://swse.deri.org/dyldo/`, last access on 12/11/2015

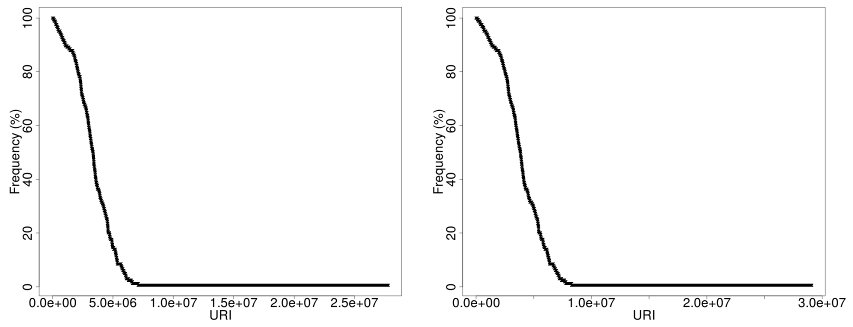


Fig. 1: Distribution of entity frequencies in the entity representation Out (left) and in the entity representation InOut (right). In both, most entities appear at only one snapshot.

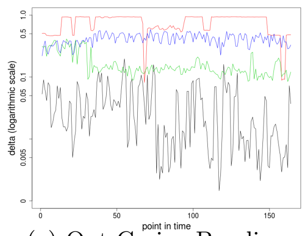
one larger than the optimized k. We introduce a cluster c_s to accommodate the static entities. In each condition, most entities belong to one cluster and other clusters have relatively small number of entities.

Figure 2 provides the representative temporal patterns of entities (i .e., centroids of observed clusters) for each condition. We observe that the number of clusters reduces significantly in conditions using information theoretic triple weighting methods. Especially, when the number of clusters is two, one cluster represents a large amount of changes consistently over time and the other shows consistent small amount of changes. Therefore, information theoretic triple weighting methods distinguish entities which have important changes from entities whose changes are less important.

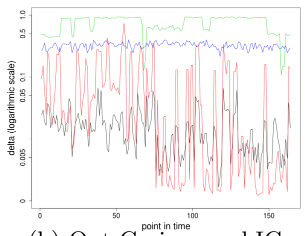
Table 4 shows periodicities of each temporal pattern shown in Figure 2. All temporal patterns have a periodicity, thus the amount of entity change has some regularity.

Table 3: Resulting clusters in each condition. The number of cluster is optimized by Average Silhouette. c_s denotes a cluster where entities have no change over all snapshots.

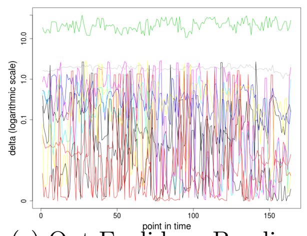
Condition			**opt. # of**	**# of entities in each cluster**				
z	**Distance**	**IC**	**clusters**	c_s	c_1	c_2	c_3	c_4
Out	Cosine	Baseline	5	1,272,764	1,205,664	9,707	26,918	6,564
Out	Cosine	combIC	5	1,349,929	1,249,998	300,043	10,637	19,157
Out	Euclidean	Baseline	11	59,700	924,104	249,598	42	214,215
					(c_5) 690,506	(c_6) 174,672	(c_7) 102,182	(c_8) 27,684
					(c_9) 28,749	(c_{10}) 50,165		
Out	Euclidean	combIC	3	0	2,491,617	30,000		
InOut	Cosine	Baseline	4	1,691,961	18,409	305,245	934,918	
InOut	Cosine	combIC	4	1,349,929	1,260,276	19,911	320,417	
InOut	Euclidean	Baseline	5	449,645	1,826,488	39,895	29,358	605,147
InOut	Euclidean	combIC	3	389,945	2,525,664	34,924		



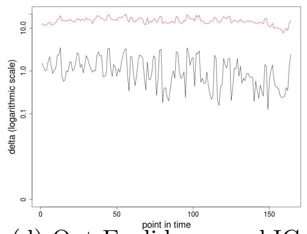
(a) Out Cosine Baseline



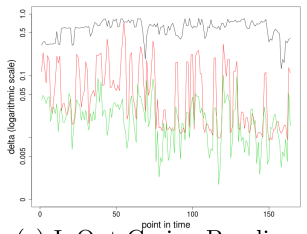
(b) Out Cosine combIC



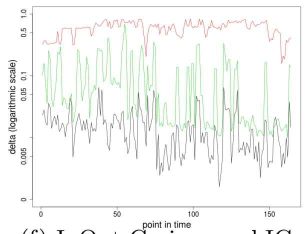
(c) Out Euclidean Baseline



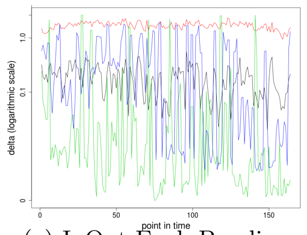
(d) Out Euclidean combIC



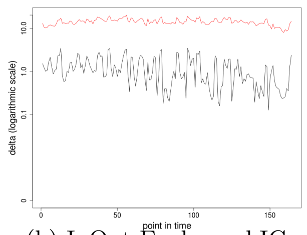
(e) InOut Cosine Baseline



(f) InOut Cosine combIC



(g) InOut Eucl. Baseline



(h) InOut Eucl. combIC

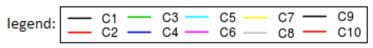


Fig. 2: Temporal patterns of entity dynamics in each condition. The x-axis denotes a point in time. The y-axis shows δ, the amount of entity changes, defined in Equations 5 and 6.

6.2 Features for Entity Dynamics

After finding the temporal patterns of entity dynamics, we investigate which features of entities more likely control the temporal patterns of entity dynamics. Umbrich et al. [23] manually analyzed entities from same domains and found out that they are more likely to have similar temporal dynamics. However, the observation is based on manual investigation and other possible features have not been empirically evaluated. In this work, we compare the following four features. Formally, we denote $f(E) \rightarrow v$ as a feature function which returns a feature value for a given entity. $v \in V_f$ is a feature value, where V_f denotes a set of all possible feature values received by f.

Table 4: Periodicity of temporal patterns of entity dynamics.

Condition			**Periodicity of each cluster (week)**									
z	**Distance**	**IC**	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
Out	Cosine	Baseline	55	16	81	29						
Out	Cosine	combIC	81	16	66	49						
Out	Euclidean	Baseline	43	77	9	69	31	31	71	25	79	18
Out	Euclidean	combIC	56	56								
InOut	Cosine	Baseline	16	55	81							
InOut	Cosine	combIC	81	16	66							
InOut	Euclidean	Baseline	55	31	79	31						
InOut	Euclidean	combIC	56	31								

- **Type (f_1)**: Most entities have one or more types defined by the predicate `http://www.w3.org/1999/02/22-rdf-syntax-ns#type` (e.g., `foaf:Person`).
- **Property (f_2)**: Predicates (e.g., `dbpedia-owl:foundedBy`) in triples describe properties of an entity.
- **Extended Characteristic Set (ECS) (f_3)**: The combination of types and properties leads to the definition of Extended Characteristic Set (ECS) [14]. Thus, in f_3, $v \in \mathcal{P}(V_{f_1} \cup V_{f_2})$. This feature function is stricter than f_1 and f_2, because while entities can have several values of f_1 and f_2, they have only one ECS.
- **Pay level domain (PLD)(f_4)**: An entity has a pay level domain (PLD) in its URI. For instance, if an entity is defined by a URI `http://dbpedia.org/resource/Facebook`, the PLD of the entity is `http://dbpedia.org`. PLD is extracted using Guava[5].

Please note that feature functions f_1 and f_2 can return more than one feature values, since entities may own several properties and types. In order to find out the features that most control the temporal patterns, we compare the clusterings generated in Section 4.3 with clusterings obtained by using the features. We denote C as a resulted clustering from Section 4.3 and $c \in C$ as a cluster.

We use Rand Index R [17] as an evaluation metric, which is defined as $R = \frac{TP+TN}{TP+TN+FP+FN}$. It is usually used to evaluate the accuracy of clustering. However, we use it to evaluate the degree of the agreement between clustering and classifications made by one of the four features. TP indicates the number of entity pairs that belong to the same cluster in C as well as that receive a same feature value v by a feature function f. For example, if the entities e_1 and e_2 belong to a same cluster and have a common feature f_4 value `dbpedia.org`, we count the entity pair as TP. TN is the number of entity pairs that belong to different clusters in C and that receive different feature values by a feature function f. The denominator denotes the number of entity pairs. Due to the computation cost, we randomly pick $10,000$ entity pairs from C and compute Rand Index.

[5] `https://github.com/google/guava/wiki/Release19`, last access on 12/17/2015

Table 5 shows the results of the analysis. In terms of f_1, f_2, and f_3, feature values of entities (i .e., properties and types) can change over time. Thus, Rand Index may differ in different snapshots. For these features, we compute Rand Index for each point in time and report mean average and standard deviation in Table 5.

Table 5: Analysis of Entity Features. For f_1, f_2, and f_3, we compute the average Rand Index over 165 snapshots, since feature values may change over time. We report standard deviation in parentheses. The largest Rand Index in each condition is marked in bold.

Condition			**Rand Index**			
z	**Distance**	**IC**	f_1	f_2	f_3	f_4
Out	Cosine	Baseline	0.4840 (0.0051)	0.5352 (0.0088)	0.5337 (0.0162)	**0.5819**
Out	Cosine	combIC	0.3994 (0.0045)	0.5814 (0.0118)	**0.6136** (0.0180)	0.5855
Out	Euclidean	Baseline	0.2334 (0.0041)	0.6890 (0.0274)	0.7766 (0.0188)	**0.8388**
Out	Euclidean	combIC	**0.9765** (0.0016)	0.2094 (0.0304)	0.0763 (0.0321)	0.1758
InOut	Cosine	Baseline	0.4397 (0.0048)	0.5585 (0.0167)	**0.5914** (0.0227)	0.5552
InOut	Cosine	combIC	0.4042 (0.0053)	0.5974 (0.0163)	**0.6241** (0.0216)	0.5891
InOut	Euclidean	Baseline	0.4485 (0.0052)	0.5673 (0.0198)	0.5646 (0.0265)	**0.5871**
InOut	Euclidean	combIC	**0.7503** (0.0043)	0.3760 (0.0286)	0.2784 (0.0158)	0.3596

7 Discussion

Periodicity of temporal patterns. In Table 4, we see that a lot of temporal patterns have periodicity of 55 and 56 weeks. It indicates that entities change on a year cycle. Thus, the amount of entity changes at a certain point in time can be predicted by looking the amount of entity changes at a year ago.

Difference between entity representations. We observe a small difference between two entity representations. A possible reason is a small difference between the numbers of triples analyzed at each entity representation. When $z = InOut$, we analyzed 16.78% triples more compared to when $z = Out$. Therefore, the effects from incoming properties is small over all.

Difference between triple weighting methods. When using Euclidean distance and combIC, we observe a large cluster where the amount of changes is consistently small (see Table 3 and Figures 2(d)(h)). It indicates that most entities are made small changes consistently. On the other hand, a small portion of entities belong to clusters where the amount of changes is large during the entire observed period. Therefore, the information-theoretic method distinguishes entities which have consistently important changes from entities whose changes are always trivial. When using the cosine distance, the difference between triple weighting methods is small. A possible reason is the normalization by the denominator in Equation 5.

Difference between distance measures. We used the cosine distance and Euclidean distance. While the cosine distance is a value between 0 and 1 (due to normalization by the denominator in Equation 5), Euclidean distance takes into account the size of entities and the number of changed triples in entities. Thus, Euclidean is a value ≥ 0. This difference influences the number of clusters. The number of clusters at conditions with Euclidean distance is always larger than when using the cosine distance. For instance, in Figure 2(c), we see a cluster where the amount of changes is always over 10 (green line). On the other hand, in the cosine distance, a value close to 1 indicates that most triples in entities are changed regardless of the number of triples.

Features for entity dynamics. When the baseline is employed as triple weighting method, the feature PLD (f_4) is most likely to determine temporal patterns of entity dynamics. This is in line with the observation by Umbrich et al.[23]. When using the cosine distance and information-theoretic triple weighting methods, the feature ECS (f_3) performs best. Since the cosine distance is proportional to the percentage of changed triples in the entities, the entity structure is important to determine temporal patterns. Thus, when using the cosine distance, it is not trivial which specific properties and types are appeared in the entities together. For instance, there is an entity with a property whose object value changes always. For this entity, the cosine distance would be small, if many of the other properties are static. In contrast, the cosine distance would be large, if the entity has only a few other properties that are static.

Potential limitation of the analysis. The analysis shown in this paper can be biased by setting the threshold to 70% appearance of entities in all of the weekly snapshots. However, we believe this bias is little, since we analyzed 71.92% of all triples that appeared in each snapshot. We think that this has grasped the most prominent and dominant temporal patterns of entity dynamics. In addition, the analysis can be biased by the dataset we used. But the dataset covers both most authoritative and randomly-selected LOD documents [11].

Co-evolution of the amount of entity changes and feature values. Feature values of f_1, f_2, and f_3 can change over time. Thus, Rand Index may differ in different snapshots. Table 5 reports the mean average and standard deviations for the index. As one can see, the standard deviations in Table 5 are overall very small. It indicates that feature values of entities are not frequently changed, as observed by Käfer et al. [10] and Gottron et al. [8]. Furthermore, the standard deviations of the feature Type (f_1) are smaller than other features (f_2 and f_3). Therefore, the feature Type f_1 is the most static entity feature among them.

Applications of this analysis. The analysis provided by this paper is useful e .g., to predict the temporal patterns and the amount of changes of newborn entities. For instance, if we get new entities, we can predict the temporal pattern of this entity dynamics by looking into feature values of entities. In terms of the information-theoretic triple weighting method, we think that they are useful to develop cache strategies for triple stores. According to [7], a predicate value is fixed in over 85% of SPARQL queries asking about a certain entity. We assume that users issue queries with more important predicates for searching entity information. However, this is subject to future research.

8 Conclusion

In this paper, we analyze the temporal patterns of entity dynamics. This analysis had not been done so far. As dataset, we use weekly LOD snapshots over a period of almost three years provided by the Dynamic Linked Data Observatory. The result demonstrates that entities that share a common PLD are more likely to change together over time, when using the baseline as triple weighting method. However, when using Euclidean distance and information-theoretic triple weighting methods, entities that have a same type are more likely to change together. Under this condition, clustering produces a large cluster where the amount of changes is consistently small and a cluster where the amount of changes is overall small. Therefore, the entity feature Type can distinguish entities that are given important changes consistently from entities that have less important changes. In terms of periodicity of observed temporal patterns of entity dynamics, most entities change on a year-cycle with small amount of changes. The result of the analysis shown in this paper can be applied to the wide range of the applications, including determining crawling strategies for LOD, caching SPARQL queries, to programming against LOD, and recommending vocabularies for reusing LOD vocabularies.

Acknowledgement This work was supported by the EU's Horizon 2020 programme under grant agreement H2020-693092 MOVING.

References

1. D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *SODA*, pages 1027–1035. SIAM, 2007.
2. L. Ding and T. Finin. Characterizing the semantic web on the web. In *ISWC*, pages 242–257. Springer, 2006.
3. R. Dividino, T. Gottron, and A. Scherp. Strategies for efficiently keeping local linked open data caches up-to-date. In *ISWC*, pages 356–373. Springer, 2015.
4. R. Dividino, T. Gottron, A. Scherp, and G. Gröner. From changes to dynamics: dynamics analysis of linked open data sources. In *PROFILES*, 2014.
5. R. Dividino, A. Scherp, G. Gröner, and T. Gottron. Change-a-LOD: does the schema on the linked data cloud change or not? In *COLD*, 2013.
6. M. G. Elfeky, W. G. Aref, and A. K. Elmagarmid. Periodicity detection in time series databases. *IEEE TKDE*, 17(7):875–887, 2005.
7. M. A. Gallego, J. D. Fernández, M. A. Martínez-Prieto, and P. de la Fuente. An empirical study of real-world SPARQL queries. In *USEWOD*, 2011.

8. Thomas Gottron and Christian Gottron. Perplexity of index models over evolving linked data. In *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, pages 161–175. Springer, 2014.
9. H. Holzmann and T. Risse. Named entity evolution analysis on Wikipedia. In *WebSci*, pages 241–242. ACM, 2014.
10. T. Käfer, A. Abdelrahman, J. Umbrich, P. O'Byrne, and A. Hogan. Observing linked data dynamics. In *ESWC*, pages 213–227. Springer, 2013.
11. T. Käfer, J. Umbrich, A. Hogan, and A. Polleres. Towards a dynamic linked data observatory. In *LDOW*. CEUR, 2012.
12. M. Konrath, T. Gottron, S. Staab, and A. Scherp. SchemEX-Efficient construction of a data catalogue by stream-based indexing of linked data. *Web Semantics*, 16:52–58, 2012.
13. M. Martin, J. Unbehauen, and S. Auer. Improving the performance of semantic web applications with SPARQL query caching. In *ESWC*. Springer, 2010.
14. T. Neumann and G. Moerkotte. Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In *ICDE*, pages 984–994. IEEE, 2011.
15. C. Nishioka and A. Scherp. Temporal patterns and periodicity of entity dynamics in the Linked Open Data cloud. In *K-CAP*, page No. 22. ACM, 2015.
16. N. Popitsch and B. Haslhofer. DSNotify–a solution for event detection and link maintenance in dynamic datasets. *Web Semantics*, 9(3):266–283, 2011.
17. W. M. Rand. Objective criteria for the evaluation of clustering methods. *J. of the American Statistical Association*, 66(336):846850, 1971.
18. P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20, 1987.
19. M. Schmachtenberg, C. Bizer, and H. Paulheim. Adoption of the linked data best practices in different topical domains. In *ISWC*, pages 245–260. Springer, 2014.
20. M. Schuhmacher and S. P. Ponzetto. Knowledge-based graph document modeling. In *WSDM*, pages 543–552. ACM, 2014.
21. C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.
22. J. Umbrich, M. Karnstedt, A. Hogan, and J. X. Parreira. Hybrid SPARQL queries: fresh vs. fast results. In *ISWC*, pages 608–624. Springer, 2012.
23. J. Umbrich, M. Karnstedt, and S. Land. Towards understanding the changing web: Mining the dynamics of linked-data sources and entities. In *KDML*, 2010.
24. X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.
25. J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *WSDM*, pages 177–186. ACM, 2011.
26. X. Yin and S. Shah. Building taxonomy of web search intents for name entity queries. In *WWW*, pages 1001–1010. ACM, 2010.