Bachelorarbeit am Institut für Informatik der Freien Universität Berlin

Human-Centered Computing (HCC)

Analyzing Behavioural Patterns in Online Knowledge Collaborations: A Case Study of Wikidata

Hong Zhu

Betreuerin und Erstgutachterin: Prof. Dr. C. Müller-Birn
Zweitgutachter: Prof. Dr. L. Prechelt

Berlin, den September 26, 2019

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den October 17, 2019

Hong Zhu

Abstract

Research into online knowledge ontologies like Wikidata has often overlooked the important interaction patterns in exploring the collaboration temporal dynamics. This thesis outlines a step-by-step procedure for investigating the frequent patterns of sequential behaviors in terms of data quality. Using a dataset of 146,450 revisions to 500 Wikidata items as a case study, the thesis employs a machine learning based tool as a proxy to evaluate the temporal item quality, and utilizes the Jensen-Shannon distance as a metric to quantify the quality variations. Referenced from the methods already developed in fields like bio-informatics, the case study mined the revisions collected using the PrefixSpan algorithm in a constraint-based fashion. Having identified the frequent sequential editing patterns that satisfy specific quality constraints, the study demonstrated the impacts of different sequence identifications regarding data quality.

Contents

List of Figures

List of Tables

1 Introduction

A knowledge-base is a technology used to store complex structured data used by a computer system. To create abundant knowledge-based goods, online knowledge collaboration, which is broadly defined as the sharing, accumulation, transformation, and co-creation of knowledge, is becoming a primary way as it allows distributed members self-organized to work for shared goals [4]. Online knowledge collaborations enormously make up the insufficiency in the traditional organizational mechanisms in consideration of the absence of stable membership, persistent interaction, or shared goals [11]. Meanwhile, its rapid development increases the complexity of structured knowledge representations, since no more single authority can develop all. Therefore, as Walk et al. [24] pointed out, it is a considerable task for us to better understand and regulate the underlying co-production processes of how users collaboratively edit the knowledge-bases.

Motivation. Existing studies investigated the structure of online knowledge collaborations often from a static perspective, yet overlooked the importance of behavioral interaction patterns in exploring the collaborative temporal dynamics. The availability of meta-data from large-scale collaborative ontology projects such as Wikidata could bridge the gap between the exploration of temporal dynamics and sequence analysis theory. A conceptualization with more emphasizes on temporal dynamics could help us better understand the complex processes in terms of contributor relationships, co-production patterns and sequential consequences. However, instead of identifying the sequential patterns in multiple dimensions, the studies are mostly restricted to a single perspective, e.g., that of contributors or activities [11].

Objective. Keegan et al. [11] proposed a general framework for analyzing multi-type and multi-stage co-production routines by relying on existing approaches to sequence analysis, followed with an empirical investigation of English Wikipedia and its communities. The goal in this work is to employ and extend the framework by specifying it into a Wikidata domain, as well as proposing conceptual and fundamental procedures to investigate *which identification of sequences is most effective in terms of data quality*. To illustrate the extended methodology as well as verify its feasibility to answer the research question, some sequence identifications were applied and investigated as a case study.

Structure. The work is organized as follows: Chapter 2 surveys related research concerning sequence analysis in online knowledge collaborations, especially in the area of Wikidata. Chapter 3 describes a methodological framework of investigating Wikidata's editing behaviours. General approaches corresponding to diverse research directions are presented explicitly in this Chapter. A case study adopting the proposed framework is conducted in Chapter 4. Chapter 5 concludes this study and discusses future elaborations.

2 Relevant Topics

2.1 Wikidata

Wikidata, known as the sibling project of Wikipedia, is a project of peer production with a high-quality, language-independent, open-licensed knowledge base. It stands out for its structured data storage and considerable application potentials. By allowing more comfortable utilization of structured data that can be presented as objective facts, Wikidata enables better data quality and higher consistency across the various Wikipedia language versions [2]. It serves as a primary data source for Wikipedia or other Wikimedia projects, and provides data to open source development projects such as OpenStreetMap[1]. Meanwhile, Wikidata increasingly demonstrated its commercial value as it has been applied to products such as Amazon smart speaker Alexa by teaching Alexa to recognize the pronunciation of song titles in different languages [20].

As an online knowledge collaboration, Wikidata allows individuals to share their knowledge in the ways that benefit the community's greater worth. Currently, Wikidata consists of $56,767,847$ content pages which are contributed by $3,278,335$ registered users[2]. Similar to Wikipedia and other socio-technical systems, Wikidata records the complete history of changes to each article since the first edit. The availability of the meta-data in forms of event logs, which describe the editing histories in detail, provides substantial opportunity for a better understanding of how the work proceeds, and how the roles are structured or transformed [11].

Like Wikipedia, Wikidata is organized in pages. Each page corresponds to an entity, which can be divided according to different namespaces mostly into an instance knowledge *item* (e.g., Berlin) or a conceptual knowledge *property* (e.g., instance of) [18]. Erxleben et al. [3] described the content model of Wikidata in detail, noting that each entity is identified by an automatically assigned unique identifier and described by the following main parts:

- Label
- Description
- Alias
- Sitelink
- Statement (Qualifier, Reference)

Label, description and alias are also known as *Term*, which is language-specific and used basically to find and display entities [3]. The sitelink connects to pages about the entity on Wikipedia and other Wikimedia projects. The statement consists of one or multiple property-value pairs (claim), and describes the entity by their characteristics [18]. It can be enriched with qualifiers and references, which provide additional context information and support. *Wikimedia Toolforge*[3], whose infrastructure is supported by a dedicated group of Wikimedia Foundation staff and volunteers, is a hosting environment for Wiki-project contributors working on services that provide value to the Wikimedia movement. These services enable contributors to do analysis easily, administer bots, run

[1] An online map with an open license, available at https://www.openstreetmap.org
[2] `https://www.wikidata.org/wiki/Special:Statistics` [Accessed: 15-Jul-2019]
[3] `https://tools.wmflabs.org` [Accessed: 28-Jul-2019]

web-services, and generally develop tools assisting other volunteers in their work. One of the critical features of Toolforge environment is the access to replications of the Wikimedia databases. After a successful application for a Toolforge developer account, all meta-data of Wikidata project can be acquired by connecting to the replication database *wikidatawiki_p* via an SSH tunnel. There are in total 83 tables storing contents about revisions, user information, and modules for development. Some primary tables like *Revision*, *User*, *Comment* provide the crucial information for analyzing the sequential editing behaviours[4].

2.2 Data Quality

As a collaborative project, Wikidata allows everyone to edit productively, which leaves it open to potentially disruptive contributions. Doubts about Wikidata's long-term viability, therefore, have been raised in terms of the quality and accuracy of statements [21]. Hence, technologies assisting in maintaining Wikidata's quality have been rapidly developed, from content changes tracking such as watch-list[5] to content changes reviewing like recent-changes stream[6]. However, a design of scalable quality control processes, for instance, automated tools for vandalism detection, is still quite necessary considering Wikidata's huge editing amount[7].

ORES (Objective Revision Evaluation Service) is a RESTful[8] API service for automated quality control, which can host machine learning classifiers for all Wikimedia projects, including Wikidata. According to Sarabadani et al. [21], it is also the first published vandalism detection classifier for Wikidata. For every edit revision, ORES generates three scores based on three machine learning models which are independent of each other. Model *damaging* is designed for reviewing potentially disruptive contributions, while model *good-faith* predicts whether an edit was saved in good-faith and identifies the potentially good-faith contributors to offer them support. As both models are used for scoring the quality of an edit, the third model *item-quality* serves primarily as a measurement of the article quality.

Different from the *damaging* and *good-faith* models for focusing on the edit behaviors, model *item-quality* puts more emphasis on the artifact, and predicts the assessment class of an article by measuring the following metrics [21]:

- Number of added/removed/changed/current site links
- Number of added/removed/changed/current labels
- Number of added/removed/changed/current descriptions
- Number of added/removed/changed/current claims
- Number of added/removed/current aliases
- Number of added/removed/current badges

[4]More detailed descriptions about these tables can be found later in Chapter 4.1.

[5]Watch-list allows editors to be notified about changes made to contents in which they are interested. `https://en.wikipedia.org/wiki/Help:Watchlist` [Accessed: 31-Jul-2019]

[6]Recent changes stream provides an interface for reviewing all changes that have been made to Wikidata. `https://www.mediawiki.org/wiki/API:Recent_changes_stream` [Accessed: 31-Jul-2019]

[7]Wikidata has a huge editing amount, according to Sarabadani et al. [21] there were about 80,000 human edits and 200,000 automated edits per day in Feb. 2016.

[8]REST (Representational State Transfer) is a software architectural style that defines a set of constraints to be used for creating Web services, RESTful Web services allow the requesting systems to access and manipulate textual representations of Web resources by using a uniform and predefined set of stateless operations. More information at `https://en.wikipedia.org/wiki/Representational_state_transfer` [Accessed: 31-Jul-2019]

Class	Criteria	Reader Experience
A	Items containing all relevant statements with solid references, complete translations, alias, sitelinks and a high quality image: • All appropriate properties for this type of item have statements with a plurality of external references for non-trivial statements, appropriate ranks and qualifiers where applicable • Translations are completed for the most relevant languages: labels and descriptions • All appropriate sitelinks to corresponding pages that exist on other wikis • All applicable aliases exist in most important languages • There is a high quality image associated with the item where applicable	All available information is recorded with reliable references
B	Items containing all of the most important statements, with good references, translations, aliases, sitelinks, and an image: • The most important properties for this type of item have statements with external references for non-trivial statements, some appropriate ranks and some qualifiers where applicable • Some important translations are completed: labels and descriptions • Most appropriate sitelinks to corresponding pages that exist on other wikis • Most applicable aliases exist in most important languages • There is an image associated with the item	All of the basic information and some extended information with references
C	Items containing most critical statements, with some references, translations, aliases, and sitelinks • The critical properties for this type of item have statements with references for some non-trivial statements • A few completed translations: labels and descriptions • Some sitelinks to corresponding pages that exist on other wikis • Applicable aliases exist only in some important languages	Most of the basic expected information is available
D	Items with some basic statements, but lacking in references, translations, and aliases • Some relevant properties for this type of item have statements • Has a label and description • Minimal applicable aliases	The statements need to provide enough information to easily identify the item
E	All items that do not match grade "D" criteria	

Table 2.1: **ORES item-quality grading scheme.**

- Number of added/removed/current qualifiers
- Number of added/removed/current internal/external[9]/unique references
- Number of changed identifiers
- Number of current complete/important translations
- Number of current important description translations
- Number of current important label translations

The model was initialized based on manual observations on 4,964 randomly selected representative articles, across Wikidata's top-level topics such as culture and art; science and technology; geography and position; people and personal life; and society, politics and history[10]. After the firsthand classification results were collected and processed as the training sample, an iterative process was carried out to train a predictable model. It employed the regression algorithms *Random Forest* and *Gradient Boosting*[11] as main approaches and, finally, achieved a prediction accuracy of 0.9707. For each revision ID as input, the model *item-quality* takes the pre-defined grading scheme (cf. Table 2.1) as a reference and returns a probability distribution of each quality class A, B, C, D, E as can be seen in Table 2.2. Each value computed presents the degree of the achievement according to the pre-defined description. As a distribution, all values are summed up to 1 and the highest probability value decides the predicted class.

As the predicted quality classes distribution reflects a temporal state of article quality, computing the item qualities of two adjacent revisions that possess a parent-child relationship, as well as comparing their respective distributions, provide an objective insight into the mechanism of quality development. Moreover, analyzing the event logs in Wikidata concerning the article quality dynamics may throw new light on studying the interactive behaviour patterns. For instance, Table 2.2 presents interesting item-quality dynamics of three serial revisions about an identical item, where the probability distributions are developing along a zigzag path. It is fascinating to figure out what exactly was done, so that the item-quality was dramatically increased for a moment and then downgraded back to the original level.

Rev ID	Parent ID	A	B	C	D	E	Class	JS	Edit Type
90357275	89561176	0.003	0.010	0.970	0.011	0.005	C	0.0	update sitelink
90389485	90357275	0.695	0.122	0.176	0.005	0.003	A	0.770	revert edit
90398504	90389485	0.012	0.073	0.921	0.003	0.002	C	0.716	remove label

Table 2.2: **Example of prediction results based on ORES item-quality model** (Each probability and JS distance is rounded up to 3 digits after the comma, the JS distance of the first revision id is coincidentally a 0).

To better understand the mechanism of quality development, it is necessary to quantify the quality variations between adjacent revisions feasibly and efficiently. The *Kullback-Leibler divergence* (KL-divergence) was introduced by Solomon Kullback and Richard Leibler in 1951 [13]. It is the most commonly used non-symmetric measure of how one discrete probability distribution Q is different from a second, reference discrete probability distribution P by calculating their relative entropy

[9] Internal references denotes the data resources from Wikimedia projects, external references refer correspondingly to data sources out of Wikimedia projects sphere.

[10] `https://www.wikidata.org/wiki/Wikidata:List_of_properties/en` [Accessed: 01-Aug-2019]

[11] `https://github.com/wikimedia/articlequality/blob/master/tuning_reports/wikidatawiki.item_quality.md` [Accessed: 01-Aug-2019]

defined as follows:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \frac{Q(x)}{P(x)}$$

where the $log(.)$ is the natural logarithm. In the simple case, a KL-divergence of 0 indicates that the two distributions in question are identical, since the divergence value grows, the difference between two distributions increases as well. Since it was put forward, it has been applied to a wide range of problems such as applied statistics, neuroscience [12] and machine learning [7]. However, despite the beautiful properties of KL-divergence, there are still some limitations that deserve more attention:

- **It is non-symmetric**. This makes it not a true metric of measuring the distance between two probability distributions, as it does not obey the triangle inequality i.e., in general $D_{KL}(P||Q) \neq D_{KL}(Q||P)$. For instance:[12]

$$D_{KL}([0.8, 0.2], [0.1, 0.9]) = 1.36273775399$$

$$D_{KL}([0.1, 0.9], [0.8, 0.2]) = 1.14572550293$$

- **It is unbounded**. Let $p(x)$, $q(x)$, $x \in X$ be two probability mass functions, then $D(p||q) \geq 0$, with equality if and only if $p(x) = q(x)$ for all $x \in X$. This makes it inappropriate for comparing distributions relatively, as the KL-divergence is only defined for distributions having compatible supports. For instance having $D_{KL}(Q||P) > D_{KL}(R||P)$ does not prove that distribution R is more similar to distribution P, compared to distribution Q.

Based on KL-divergence, the *Jensen–Shannon divergence* (JS-divergence, also known as information radius or total divergence to the average) is a method in information theory for measuring the similarity between two probability distributions, defined as follows:

$$D_{JS}(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M)$$

where $M = \frac{1}{2}(P + Q)$. Unlike the Kullback-Leibler divergence, it is symmetric, always well-defined, and most importantly, bounded. With the $log(.)$ denoting base 2 logarithm, the JS-divergence is bounded as follows [15]:

$$0 \leq D_{JS}(P||Q) \leq 1$$

A *Jensen-Shannon distance* is generated by computing the square root of the JS-divergence, which can be viewed as a metric of measuring the distance between distribution P and Q:

$$\sqrt{\frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M)}$$

Therefore, the JS-distance provides an optimal opportunity for quantifying the article quality dynamics, as the *item-quality* model returns discrete probability distributions as output, and it satisfies the applied condition to compute the divergence value. A local approach to quantify the article changes is calculating all distances, i.e., JS-distance values of each adjacent revision's item quality distribution $D_{JS}(P_{parent\_rev}||Q_{rev})$, which works as a sliding window. Table 2.2 presents the JS-distance value between each revision and its antecedent. As the table revealed, the item quality

[12] Results are calculated with help of the Python Scipy library: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.entropy.html` [Accessed: 13-Sep-2019]

was vastly improved from class C to A after a potential vandalism revision being reverted, only it did not last long. Having quantified the quality variations into JS distances, it is easy to see that the two revisions are not dragging the quality up and down to the same extent, which implies that the third revision, where a label was removed, is potentially more effective and powerful.

2.3 Sequence Pattern Mining

The problem of (frequent) sequential pattern mining was firstly addressed and defined by Agrawal and Srikant [1] in 1995 as follows:

Given a database of sequences, where each sequence consists of a list of transactions ordered by transaction time and each transaction is a set of items, sequential pattern mining is to discover all sequential patterns with a user-specified minimum support, where the support of a pattern is the number of data-sequences that contain the pattern.

It is about discovering all frequent sequential patterns according to the number of sequences that contain these patterns by giving a collection of chronologically ordered sequences [10]. As all events, subsequences or substructures that frequently appear in a data-set would be captured, this problem is described succinctly by Masseglia [16] as the discovery of all temporal relations between facts embedded in a database.

More formally, the problem of mining sequential patterns along with its associated notation is defined as follows (cf. [17]): Let $I = \{i_1, i_2, ..., i_m\}$ be a set of literals termed *items*, an *event* $E = (i_1, i_2, ..., i_k), i_j \in I$ is a non-empty unordered collection of items, A *sequence* $\alpha =< \alpha_1\ \alpha_2\ ...\ \alpha_n >$, $\alpha_i \in E$ is an ordered list of events, where a sequence with k-items is a *k-sequence* if $\sum_j |\alpha_j| = k$. Table 2.3 presents the sequence database construction according to the existing event logs. All activities belonging to the same artifact are assigned into one identical sequence, where the consecutive edits committed by the same user are collapsed into one identical event. Furthermore, a sequence $< \alpha_1\ \alpha_2\ ...\ \alpha_n >$ is a *sub-sequence* of another sequence $< \beta_1\ \beta_2\ ...\ \beta_m >$, i.e. sequence $< \beta_1\ \beta_2\ ...\ \beta_m >$ is a *super-sequence* of $< \alpha_1\ \alpha_2\ ...\ \alpha_n >$, if there exist integers $i_1 < i_2 ... < i_n$, so that $\alpha_1 \subseteq \beta_{i1}, \alpha_2 \subseteq \beta_{i2}, ..., \alpha_n \subseteq \beta_{in}$. For instance, the 3-sequence $< (B)(AD) >$ is a sub-sequence of 7-sequence $< (AB)(E)(ABCD) >$. Given a database $D = \{\alpha_1, \alpha_2 ... \alpha_n\}$, the problem of sequential pattern mining is about discovering all frequent sequences α and their sub-sequences whose frequencies of occurrence are no less than a user-specified threshold *min_ support*. In addition, there are two essential concepts proposed to improve the effectiveness of results mined. A pattern α is a *closed frequent pattern* in D if α is frequent and there exists no proper super-sequence β, so that β has the same support as α in D. And a pattern α is a *maximal frequent pattern* in D if α is frequent and there exists no super-sequence β, so that $\alpha \subset \beta$ and β is frequent in D. According to Han et al. [10], the set of closed frequent patterns is more compact and contains the complete information regarding its corresponding frequent patterns given the same minimum support, which is a nice option for mining high density data.

Since sequential pattern mining was first addressed by Agrawal and Srikant [1], there have been extensive studies on the improvements or extensions about the mining algorithms. The most representative algorithms for mining sequential patterns can be generally divided into the following classes:

Apriori-based algorithms. Agrawal and Srikant [1] discovered an interesting fact that among frequent k-sequences, *"a k-sequence is frequent only if all of its sub-sequences are frequent"*. They

Activity (edit)	**Artifact** (article)	**Performer** (user-id)	**Order** (timestamp)
set alias	Q30	69382	03-08-08:34
update description	Q30	69382	03-08-12:47
add alias, remove alias	Q30	22594	03-11-01:39
set label	Q30	22594	03-17-00:16
set sitelink	Q30	146231	03-17-00:18
add claim	Q31	13	03-18-18:00
update claim	Q31	13	03-18-18:20
add description	Q31	13	03-18-18:21
update claim	Q31	13606	03-18-18:59

Seq. ID	**Edit Sequence**
Q30	<(set alias, update description) (add alias, remove alias, set label) (set sitelink)>
Q31	<(add claim, update claim, add description) (update claim)>

Table 2.3: **Example of Wikidata event log sequences in horizontal formatting data layout** (data layout is adapted from Agrawal et al. [1]).

termed this downward closure property as *Apriori*. This implies the *AprioriAll* algorithm that frequent sequences can be mined by scanning firstly the sequence database to obtain the frequent 1-sequences, based on which frequent 2-sequences candidates are generated. After that all candidates are checked against the database to find the final frequent 2-sequences, and the whole process repeats until no more new frequent k-sequences can be discovered [10]. Briefly, AprioriAll is a three-phase algorithm consisting of finding all $k-1$ sequences satisfying the minimum support, generating k sequence candidates with possible pruning. In addition to the AprioriAll algorithm, *Generalized Sequential Patterns* (GSP) is also a representative of Apriori-based sequential pattern mining algorithm, which generalized the earlier notation of AprioriAll algorithm to time constraints which specify a minimum or maximum time period between each adjacent element, sliding time window within which the events occurred, and user-defined taxonomies allowing sequential patterns to include items across various levels [22].

SPADE algorithm. The *Sequential PAttern Discovery using Equivalence classes* (SPADE) algorithm was firstly proposed by Zaki [25] in 2001 for solving the problems of making repeated database scans in Apriori-like algorithms. It creatively uses a vertical id-list database format, where a list of objects can be associated to each occurred sequence. After that, the frequent sequences can be efficiently found by using intersections on id-lists. This algorithm deconstructs the original problem into smaller sub-problems, uses efficient lattice search techniques to solve them in main-memory, significantly reduces the number of databases scans, and therefore also reduces the execution time [25].

Pattern growth algorithms. The *Frequent-Pattern growth* (FP-growth) algorithm proposed by Han et al. [8] is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth. It uses an extended prefix-tree structure for storing the compressed information about frequent patterns named *Frequent-Pattern tree* (FP-tree). Based on this basic idea, Han et al. [9] proposed the *FreeSpan* algorithm, which uses frequent items to recursively

project the sequence database into a set of smaller projected databases, and let the sub-sequence fragments iteratively grow in each projected database. Later, based on FreeSpan, Pei et al. [19] developed the *PrefixSpan* algorithm, which examines only the prefix sub-sequences and project only their corresponding postfix sub-sequences into the projected database[13].

[13]This approach will be further explained in detail in Chapter 4.3.

3 Sequence Analysis Methodology

Previous research has already demonstrated the applicability of discovering temporal dynamics embedded in the online knowledge collaborations by mining and analyzing the sequences. Keegan et al. [11] explored a particular aspect in co-production routines to yield insights regarding the number of contributors participating in the article co-creation process by analyzing the sequential event logs. Similarly, Cuong et al. [2] discussed the feasibility of discovering ponderable role transitions, dominant dynamic participation patterns as well as statistical inferences deductions in Wikidata inferred by constructing sequence analysis.

According to Keegan's [11] proposal, the fundamental event log features that constructing sequences should be employed to capture the temporal relations, as they allow themselves to be decoded into the messages in a *who-what-when-how* architecture, which is defined as follows:

Performer. An entity executing an activity such as a user. This describes who is responsible for a specified revision. Depending on different performer characterizations or how a performer participates in the co-production process, a performer can be observed from the following perspectives:

- **User Group**: as Keegan et al. [11] pointed out, performers are not only single anonymous human users, but potentially: sub-users with an account, e.g., a human with developing tools, or automated users, e.g., bots. These, moreover, could be further classified as bots with requests and bots without requests[1].
- **Role of Performer**: a performer may possess one or more particular roles corresponding to different user rights such as interface-administrator, rollbacker or IP block-exempt.

Artifact. An entity in Wikidata which is operated upon, such as an item, describes what has been edited.

- **Name Space**: assign each article to its respective namespace[2]. For instance, an article is identified as an Wikidata item if it has a namespace ID 0, or a Wikidata property for possessing a namespace ID 120.
- **Domain**: assign each article to a specific domain according to its class. For instance, Färber et al. [5] proposed a novel evaluation of determining the coverage of domains concerning the classes per knowledge graph by manually assigning the most used classes into the following five domains: people, media, organizations, geography and biology.
- **Development Stage**: how an article is progressing and to what extent it has been completed. Each article had an initial quality level E when it is created (cf. Chapter 2.2), and as the edits increase, consequently the article quality changes and on the whole presents a growth trend. Therefore, the current quality level of an article indicates its development stage.

[1] According to Wikidata's policy system, every (semi-)automatic task carried out by a bot needs to be approved by the community. It means that operators must open a request for permissions for their bots before they can run them on Wikidata

[2] Name spaces allow for the organization and separation of content pages from administration pages, and each article is assigned to exactly one name space. More information at `https://www.wikidata.org/wiki/Help:Namespaces` [Accessed: 21-Aug-2019]

Performer (user-id)	**Artifact** (article)	**Activity** (edit-type)	**Order** (timestamp)
13	Q17	create page	2012-10-29, 17:20
3102	Q17	set label	2012-10-29, 18:05
2	Q17	set sitelink	2012-10-29, 18:28
2936	Q17	set alias	2012-10-29, 19:02

Table 3.1: **Example of an Wikidata event log.**

Order. An index defining a sequence such as a timestamp, this describes when a specific revision was documented, and enables us to construct the temporal relations by restricting each adjacent event to an explicit processor-successor relationship.

Activity. A system action such as a revision to an article, this describes how an entity has been edited. Traditionally, in the Wikidata domain, an activity is recorded entirely in the form of comments, based on which, according to different research directions, an activity can be extended into the following sub-features[3]:

- **Edit Summary**: a structural activity pattern that is extracted from the original activity description recorded in the database, for instance, *wbsetsitelink-set* is an edit summary extraction from the HTML markups */*wbsetsitelink-set:1/dewiki*/Japan*
- **Edit Type**: what kind of editing has been done in a regular and structural format e.g. add claims.

3.1 Identification of Sequences

The identification of sequences is about the selecting, structuring and combining of aforementioned features, i.e., dimensions into event logs that set the stage for the subsequent work. A Wikidata event log, featured with some of the dimensions, describes an integrated collaboration across performers. As can be seen in Table 3.1, each horizontal row allows itself to be decoded into a *who-what-how-when* message, while the whole event log can be regarded as a sub-sequence in terms of sequential patterns with a vertical perspective, as it is single artifact related, i.e., no artifact variation. Based on the four fundamental features according to Keegan et al. [11], there are $\sum_{i=1}^{4} \binom{4}{i} = 15$ possible different feature combinations, upon which an event log of multiple dimensions is constructed. Under particular circumstances, each feature can be further extended into above mentioned sub-features which makes this figure rise up, as shown in Table 3.2. Having constructed different sequence identifications by combining various event log features, effectiveness of sequential patterns discovered could be further evaluated in terms of data quality.

3.2 Schematization of Sequences

Once the sequences have been identified, it is necessary to formulate the most relevant elements, i.e., event log features regarding a particular research aspect into a knowledge representation scheme [11], where the chosen features are expanded as detailed records, either according to the official

[3] An explanation about how is a comment extended concretely into the sub-features can be found in Chapter 3.2

	Activity	Artifact	Performer	Order
Activity				
Artifact				
Performer				
Order				

Activity	Edit Summary Edit Type
Artifact	Namespace Development Stage Domain
Performer	User Group Role of Performer
Order	Time Range

Table 3.2: **Example of two dimensional feature selections.**

documentation or based on the researcher's own code book. Lazar et al. [14] investigated the necessity of data pre-processing for statistical analysis. As they pointed out, the data collected may be presented in inconsistent formats that needs first to be filtered and fixed in case of data contamination. Furthermore, the original data may be too primitive for the readers to identify the underlying themes. Taking one of the fundamental features, *activity* for instance, the pre-processing steps are according to Lazar et al. [14] defined as follows:

Data Cleaning. It is an essential step to clean up the data due to inappropriate formatting or descriptions. For each revision in Wikidata, there is typically a textual comment describing its action by summarizing the change, e.g., an editor's edit summary. Below are some representative revision comments:

- */*wbsetsitelink-set:1|dewiki*/Japan*
- */*wbcreateclaim-create:1|*/[[Property:P530]]:[[Q29999]],#distributed-game*
- */*wbsetsitelink-add:1|zhwikiquote*/*圆周率

The part between */* */* indicates which Wikibase API module (after the first */** symbol)[4] has been called by which operation (after the short hyphen) and from which source project the data has been generated (after the long hyphen), while the latter half of comment (after the **/ symbol* describes the detailed edit content. In addition, the tools that have been used to edit the items are documented after the occasionally appearing # symbol. Though the original comments collected are documented in a relative structural and consistent format, they are still required to be processed and aggregated given their high primitiveness. More importantly, more attentions need to be paid to the summarized editing behaviours such as edit types, rather than the concrete contents which are relatively irrelevant to our research question. The regular expressions, which are often used to mean the specific, standard textual syntax for representing patterns of matching text, are therefore employed to locate, capture and extract the first part of */*...:*, and which are, taking again the aforementioned comments, for instance:

- *wbsetsitelink-set*
- *wbcreateclaim-create*

[4]Wikibase provides a general mechanism to store statements as structured data in Wikidata. The Wikibase API is provided by a set of extensions that implement MediaWiki API modules, which is a mature and stable interface that is actively supported and constantly improved. See `https://www.wikidata.org/w/api.php?action=help&modules=main` [Accessed: 09-Jul-2019]

Edit Type	Edit Summary	Paraphrase
Add alias	wbsetaliases-add	Add new aliases to the existing alias list of a Wikibase entity, won't overwrite
Set alias	wbsetaliases-set	Set a new alias list for a Wikibase entity, possibly overwrite
Update alias	wbsetaliases-update	Update the alias list of a Wikibase entity
Remove alias	wbsetaliases-remove	Remove aliases from the existing alias list of a Wikibase entity
Update alias	wbsetaliases-add-remove	Add new aliases to the alias list as well as remove aliases from the list

Table 3.3: **Edit type coding scheme of Wikibase API *wbsetaliases* module.**

- *wbsetsitelink-add*

Data Coding. Though the original comments are highly structured into edit summaries containing information of the API, there are still some unambiguous parts that need to be further processed, e.g., a clear distinction between *setsitelink-set* and *setsitelink-add*. Therefore, developing representative descriptions of edit summaries with help of the content analysis is necessary before the statistical analysis can be conducted. Content analysis, as Stemler [23] stated *"is a systematic, applicable technique for compressing many words of text into fewer content categories based on explicit rules of coding."*, which enables researchers to *"sift through large volumes of data with relative ease in a systematic fashion."* The coding process is to analyze the content of a text by assigning categories as well as descriptions to the text. According to Stemler [23], this process can be divided into *Emergent coding* and *a Priori coding*.

Different from emergent coding, where the text content is needed to be more broadly understood without any particular starting point, analyzing the content with a priori coding approach is commonly based on theoretical frameworks, i.e., taxonomies, which guide the researchers in understanding the data collected through existing contributions. This approach becomes our choice for schematizing edit summaries, as coding the edit content in Wikidata has already been conducted in previous studies. Müller-Birn et al. [18] investigated the contextual information about edit comments in detail and applied several steps to identify the edit types, where the edit comments are categorized and aligned into consistent verb-noun pairs that indicate which actions have edited which content model.

Furthermore, ensuring the reliability of qualitative coding is a crucial yet challenging task in terms of coding objectivity as a series of decisions regarding text interpretations are often made by human researchers. Hence, all related information should be taken into account to assign each raw data into the designed codes. For instance, the edit summary contains the module information as well as the operations being called, therefore the API documentation of MediaWiki would potentially be a primary reference for understanding the edit summary context. As can be seen in Table 3.3, information retrieved from the MediaWiki API is documented under the paraphrase column as an interpretation to edit summaries, depending on which the edit types are coded. This ensures the evolving nature of coding scheme construction, as the organization of scheme components expands and constantly changes during the coding process.

3.3 Analysis of Sequences

After the extracted sequences are schematized, quantitative methods for analyzing sequential behaviors will be employed to capture the most representative editing behaviors. Keegan et al. [11] classified the following three categories of quantitative approach according to various study focuses:

- **Pattern mining**, using intersected methods of statistics, data base systems and machine learning to discover frequent patterns of different sequences in large data sets.
- **Sequence similarity**, which provides a reliable, applicable strategy for characterizing the newly determined sequences.
- **Probability analysis**, some stochastic models like *Markov chain* are used for sequence predictions under a specific probability, where a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

As one of the most representative methods for analyzing sequential behaviours, sequence pattern mining is about discovering all sequential patterns weighted according to the number of sequences that contain these patterns [24] (cf. Chapter 2.3). Having employed the sequence mining algorithms, quantitative investigations can be conducted in order to discover the actual frequent patterns responsible for the regularities in the event logs.

Sequence ID	Editor ID	Timestamp	Edit Type	JS Distance
114	56389	10-29-17:30	set sitelink	0.0181
114	56389	10-29-17:31	set sitelink	0.0037
114	3232667	10-29-17:44	set sitelink	0.0216
114	780221	10-29-18:05	set label	0.0037
114	25876	10-29-18:13	set sitelink	0.0004
114	25876	10-29-18:14	set label	0.0439
114	4251	10-30-15:42	set description	0.0310
114	4251	10-30-15:43	set sitelink	0.0012

State	Activity Sequence
Before Pruning	<(set sitelink, set sitelink) (set sitelink) (set label) (set sitelink, set label) (set description, set sitelink)>
After Pruning	<(set sitelink) (set sitelink) (set label) (set description)>

Table 3.4: **Example of constraint-based pruning with the JS-distance greater than** 0.01**.**

To investigate *which identification of sequences is most effective in terms of data quality*, one of the most significant yet challenging tasks is to integrate the topic of sequential pattern mining

with data quality, i.e., to conduct the mining process with a data quality perspective. However, the conventional mining systems provide only a very restricted mechanism, primarily the minimum support for specifying patterns of interest. Garofalakis et al. [6] argues in their work the significance of user-controlled focus to be incorporated in the pattern mining process. As they pointed out, without taking user-specified constraints into consideration, the system may execute an appropriate mining algorithm and return a large number of sequences, most of which the users are not interested in at all. Therefore, some mechanisms like *Constraint-based mining* can be used to deal with the process, which efficiently mine only the patterns satisfying the user-specified constraints. *Succinct constraints*, as one of the representative constraints, can be pushed into the initial data selection process at the start of mining, which significantly reduces the size of original dataset [10]. Garofalakis et al. [6] explored the context of sequential pattern mining by setting a flexible, user-specified regular expression constraint for selecting the data to be mined. This was used in their work for mining of the World Wide Web (WWW) user access logs and determined the most frequently accessed topic paths.

Inspired by their work, this study conducts the mining process in a constraint-based fashion, where the aforementioned JS-distance (cf. Chapter 2.2) would be employed as a succinct constraint to be pushed at the start of mining. As discussed, the finite property and symmetrical characteristic of JS distance make it an optimal metric for measuring the quality dynamics. Table 3.4 shows an example of the initial data selection with a JS-distance constraint of 0.01, where all event logs not satisfying this constraint (below 0.01) are pruned before the activity sequences are constructed. It is worth noting that the temporal order, which is the cornerstone of mining sequential patterns, is preserved as the remaining event logs still hold the processor-successor relationships.

3.4 Interpretation of Sequences

As the final step, this work in this phase is about enriching the results obtained through previous quantitative investigations by employing qualitative methods. In order to obtain various patterns to be used for evaluating the effectiveness, the data collected will be mined by adjusting the user-specified constraints, which in our case are the *JS distance* and *minimum support* thresholds. As a core part to be investigated in this phase, once the patterns are obtained, different metrics for measuring the effectiveness will be employed to get a better observation and evaluation, such as:

- Frequency of the pattern
- Length of the pattern
- Diversity of the pattern set in terms of edit focus and activity type

The observed patterns will be further illustrated by applying the descriptive statistics on sequences provided and analyzing the contextual factors that may affect the results.

4 Case Study

In this case study, an empirical investigation of the quality dynamics in Wikidata and the corresponding editing behaviours will be conducted using sequential pattern mining. This case study is a detailed examination of one or more specific situations and can be seen as a pilot study for investigating the effectiveness of sequence identifications in terms of data quality. As discussed before, the identification of sequences is primarily about the selecting, structuring and combining of the event log features (cf. Chapter 3.1). Therefore, the case study will focus on the features of *Activity* and *Artifact*, representing activity in the forms of edit type and expending artifact into sub-features namespace and development stages (cf. Table 4.1).

	Activity	**Artifact**	**Performer**	**Order**
Activity				
Artifact	×			
Performer				
Order				

Activity	Edit Summary
	Edit Type
Artifact	Namespace
	Development Stage
	Domain
Performer	User Group
	Role of Performer
Order	Time Range

Table 4.1: **Two dimensional feature selections for the case study.**

4.1 Sample and Data Extraction

Using the aforementioned Wikidata Toolforge database *wikidatawiki_p* (cf. Chapter 2.1), a sample including 500 randomly chosen articles is collected. All articles are restricted to the namespace 0 and 120 (items and properties) as they represent a structured dataset that consists of conceptual and instance knowledge, which Wikidata stands for [18]. In order to investigate the correlation between data quality and the development stage of artifacts, for each ORES quality class i among A, B, C, D and E, the dataset is determined in the following two steps:

1. Get all item numbers for which the current quality level is i.
2. Randomly select 100 items by using a random generator.

In general, the entire dataset was selected based on randomization and covers diverse Wikidata topical categories including culture, science, geography, people and history. Figure 4.1 presents the category distribution in detail, and for each item the content domain is classified according to its English label, description and values of the *instance of* property, which is extracted by querying in *Wikidata Query Service*[1]. As the chart shows, the distribution is relatively balanced across different category clusters, which makes it representative as the results can be less influenced by

[1]The Wikidata Query Service provides a way for tools to access Wikidata data, via a SPARQL API. See `https://www.mediawiki.org/wiki/Wikidata_Query_Service` and `https://query.wikidata.org` [Accessed: 20-Jul-2019]

item domains. There are primarily three tables in the *wikidatawiki_p* database (cf. Chapter 2.1) used for collecting the information corresponding articles and activities, which is essential for mining the interaction patterns with a data quality perspective:

- **Page**[2]. Considered as the "core of the wiki", each page stored in the page table has an entry which is identified by the title and contains some essential meta-data.
- **Revision**[3]. This table contains significant meta-data for every edit done to a page, especially the information about who made the edit and at which time the edit was made.
- **Comment**[4]. The edits, blocks and other actions for each revision are described as a textual comment and stored in this table, which makes it a primary source for extracting concrete edit information.

According to the tables above, all activities in the sample are tracked from each article's inception until March 31, 2019. In addition to the Toolforge database, ORES API (cf. Chapter 2.2) is used as a primary source to capture the article quality. On the basis of revision ids, quality results are recorded as a series of discrete distributions, representing quality level classifications (cf. Table 2.1). As the investigation focuses on the influence of edit activities to article quality dynamics, 3410 revisions without any comment records and 5 revisions, of which the comments are semantically ambiguous, are excluded. This results in a dataset including 146,450 editing activities of 499 item articles made by 5,756 distinct contributors.

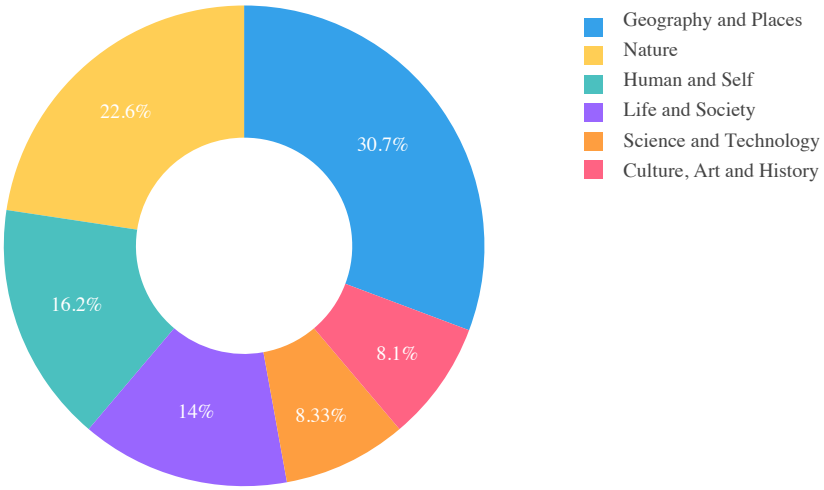


Figure 4.1: **Topical categories overview of items in the data sample** ($n = 420$, the 80 items without an English label are excluded).

[2] `https://www.mediawiki.org/wiki/Manual:Page_table` [Accessed: 21-Jul-2019]
[3] `https://www.mediawiki.org/wiki/Manual:Revision_table` [Accessed: 21-Jul-2019]
[4] `https://www.mediawiki.org/wiki/Manual:Comment_table` [Accessed: 21-Jul-2019]

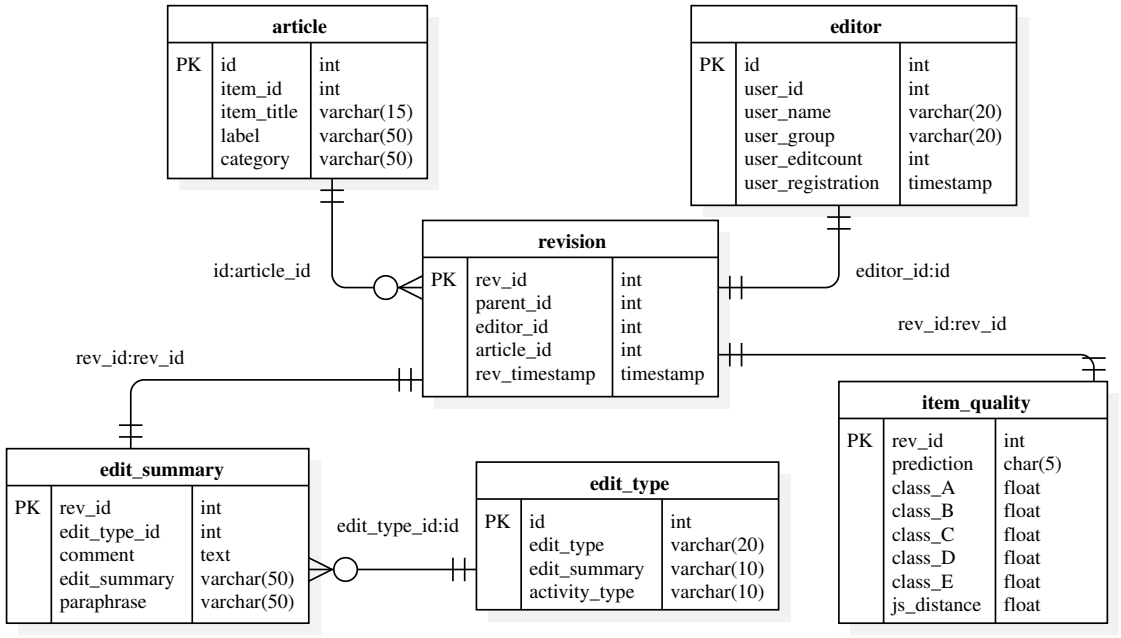


Figure 4.2: **Database schema of article sample in an ER-diagram format.**

4.2 Data Representation

In order to analyze the correlation between sequences of activities and article quality dynamics, it is necessary to devise a consistent way for representing the data. As described above (cf. Chapter 3.2), the collected raw data of activity edit records, i.e., comments, are first extracted into edit summaries, then collapsed on a basis of semantic categorization to aggregated edit types. The edit types are aligned in consistent verb-noun pairs, and represent content models *edit focus* and concrete edit actions *activity type*. Consequently, 54 edit summaries are aggregated into 35 distinct edit types, regarding 10 edit focuses and 10 activity types as schematized in Appendix 5.1.

After the data collection and processing, the case study creates a relational SQLite database to represent and store the data[5], which is capable of handling abundant complicated queries, database transactions and routine analyses. Figure 4.2 presents the database schema in an ER-diagram: the substance table *revision* is the central table, which connects all properties and composites representing the basic *who-when-what-how* event log features, while the branch tables connected to the centre are responsible for expanding the core features into their respective sub-features. By conducting SQL queries[6], activity sequences could be efficiently built with desired identifications. However, as a previous step, a number of descriptive statistical tests are being conducted to better understand the nature of the dataset, as they provide essential guidelines for identifying the sequences, and, in particular, interpreting the later resulting patterns.

Table 4.2 presents a few descriptive statistics in terms of JS-distance value in the sample data. As can be seen from the results, although the quality variation triggered by a single edit activity could be up to 0.9879, which, to some extent, can be considered as the power of this edit to affect

[5]The database dump is published and located in: `https://github.com/Felihong/wikidata-sequence-analysis/tree/master/data`.

[6]SQL is a standardized computer language for relational databases, addressing functionality such as searching for, updating, creating and deleting data.

[7]Here refers to the revisions, of which the JS distances are not calculated as 0.

	Average	Maximum	Minimum
All	0.0098 ($\sigma^2 = 0.003$)	0.9879	0.0
Significant Only[7]	0.0197 ($\sigma^2 = 0.006$)	0.9879	0.0

Table 4.2: **Descriptive statistics of JS distance in the data sample** ($n = 146,450$, all JS values are rounded up to 4 digits after the comma, the original minimum JS value of significant revisions is 0.000000020022106750032484).

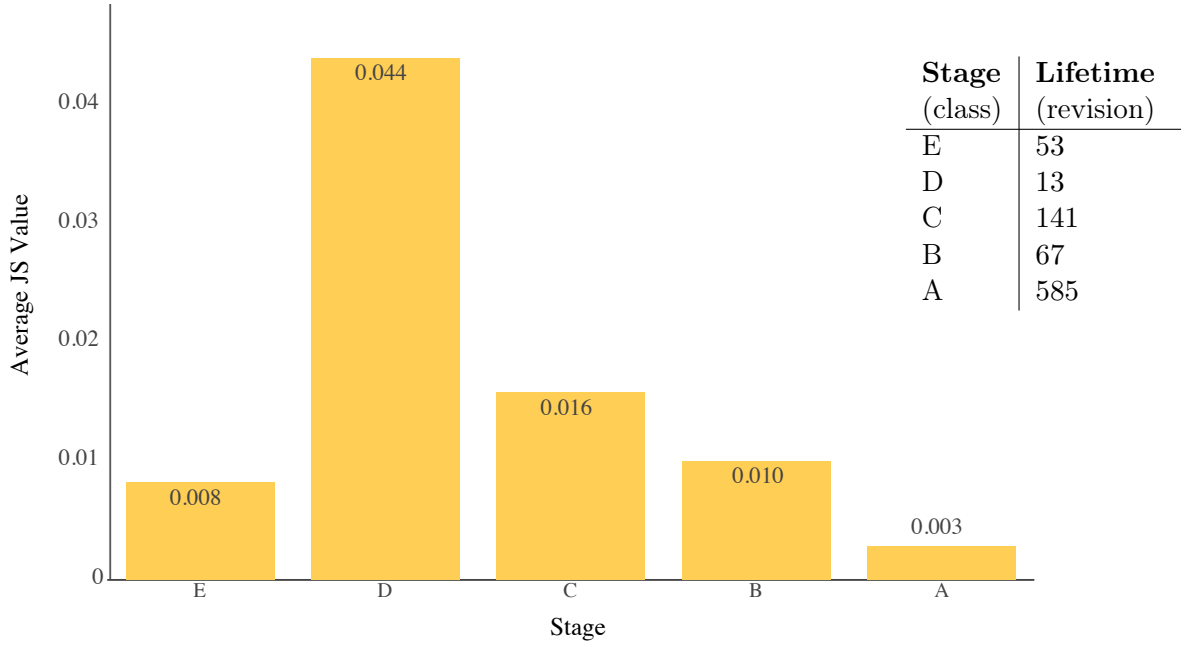
the article quality, the average quality variation is rather small (0.0098). This is not surprising, given that the vast majority (80.74%) of JS-distances are less than 0.005.

Moreover, it is noticeable that a half (50, 56%) of editing activities are not considered as responsible for the quality changes: There are 74, 046 revisions of the whole dataset (146, 450) holding a JS distance value of 0, among which the predicted quality distribution between each adjacent edits is identical. This indicates that the article quality variations of these revisions are not significant enough to be captured by ORES classifier, or, in other words, the sensitivity of ORES item-quality model is yet to be improved enough to detect all subtle changes, as the involved edit focuses are covered in the model training features (cf. Chapter 2.2).

Edit Focus	Edit Type	Item	Content Variation	Quality Variation
Alias	set alias	Q698	`https://is.gd/JJttJA`	`https://is.gd/geWFXM`
	add alias	Q312	`https://is.gd/irLaJD`	`https://is.gd/42vhvL`
	remove alias	Q198	`https://is.gd/lNKPIa`	`https://is.gd/hdUaPS`
Claim	set claim	Q34	`https://is.gd/evgxPQ`	`https://is.gd/92Y8JN`
	add claim	Q71	`https://is.gd/2Qd1Om`	`https://is.gd/8GczcT`
	remove claim	Q526	`https://is.gd/8RvPOk`	`https://is.gd/tX3ocn`
Description	set description	Q4759	`https://is.gd/NYVdM6`	`https://is.gd/z9IBsS`
	remove description	Q2	`https://is.gd/r0mtam`	`https://is.gd/HSRxI7`
Edits	revert edits	Q355	`https://is.gd/DtNIZg`	`https://is.gd/Jtqd1y`
Item	set item	Q17	`https://is.gd/eyeGom`	`https://is.gd/Dioa2I`
	merge item	Q22	`https://is.gd/zK7uE2`	`https://is.gd/hZtQvB`
	update item	Q747	`https://is.gd/ST32Lv`	`https://is.gd/KSWlRI`
	protect item	Q312	`https://is.gd/zeiF0Q`	`https://is.gd/oacVsy`
Reference	set reference	Q549	`https://is.gd/ZEIpfz`	`https://is.gd/vjHCJV`
	add reference	Q633	`https://is.gd/qvpnpI`	`https://is.gd/5dJasC`
	remove reference	Q183	`https://is.gd/aYjOO1`	`https://is.gd/16d3T2`
Sitelink	set sitelink	Q30	`https://is.gd/XA3iYd`	`https://is.gd/W3eCQp`
	update sitelink	Q22	`https://is.gd/5nH6yU`	`https://is.gd/jY4JTD`
	remove sitelink	Q377	`https://is.gd/J8r0xC`	`https://is.gd/pE39Ux`
Term	remove term	Q83	`https://is.gd/i1wI1b`	`https://is.gd/GKrCxa`

Table 4.3: **Example of undetected article quality variations** ($n = 20$, the URLs of difference pages are shortened due to the length limit[8], the list is ordered alphabetically according to the edit focuses).

To further verify this observation, this case study conducted an investigation by reviewing the article differences before and after a particular edit activity, and observing the triggered quality variations. For each typical edit type among the non-significant revisions, one representative revision is randomly chosen to be compared with its preceding revision by reviewing Wikidata's page history and the quality distributions predicted by ORES (cf. Table 4.3). Remarkable changes can be seen by filtering out the non-significant revisions, that is to say, the revisions calculated with JS distances of 0. As shown in Table 4.2, the average value of quality variation is increased to 0.0197 by ignoring the non-significant revisions, leaving 30.18% of revisions holding a JS-distance less than 0.005.



Stage (class)	**Lifetime** (revision)
E	53
D	13
C	141
B	67
A	585

Figure 4.3: **The average JS distance value and lifetime over each article development stage** ($n = 146,450$, each JS distance is rounded up to 3 digits after the comma, the life time is measured by the average number of revisions at each developing stage).

A more telling statistic calculates the quality variation with a dynamic perspective. It is shown that the degree of quality variation in an article is closely related to the article's current development stage (cf. Chapter 3). As can be seen in Figure 4.3, the average quality variation is quite different at each development stage. Surprisingly, developing articles in their D stage appear more susceptible than any other time, as the quantified quality variations significantly outnumbered the other stages (0.044). On the contrary, articles at the A stage are considered as robust, as the quality variation triggered by a single edit is merely 0.003. This also explained the long lifetime of articles at this stage ($avg = 585, revisions$) compared to the brief stopping at the phase D ($avg = 13, revisions$).

[8] `https://is.gd/index.php` [Accessed: 03-Jul-2019]

Overall, the power of a single edit to affect the article quality becomes weaker as the articles become more and more complete.

Table 4.4 below presents the most significant (highest JS distance value on average) single edit types. While few of the significant edit types are highly common, it is noticeable that all edit types concerning edit focus *edits* are listed (cf. Appendix 5.1), where the edits are detected either as vandalisms or improper reverts of vandalisms so that are rolled-back to the preceding version. Moreover, all edit types in regard to edit focus *reference* appear in the list as well, which reflects the large proportion placed on reference in the training features (cf. Chapter 2.2). To get a deeper understanding of the factors affecting quality development, the correlation between quality change and edit types is further investigated from the following perspectives: edit focus and activity type.

Edit Type	**Count**	**%**	**Average**	**Maximum**	**Minimum**
unrevert edits	1	n/a	0.672 ($\sigma^2 = 0$)	0.672	0.672
set reference	1,486	1.01%	0.136 ($\sigma^2 = 0.543$)	0.901	0.0
merge item	512	0.35%	0.055 ($\sigma^2 = 0.024$)	0.988	0.0
remove reference	311	0.21%	0.051 ($\sigma^2 = 0.013$)	0.865	0.0
set term	314	0.21%	0.043 ($\sigma^2 = 0.006$)	0.879	0.0
remove description	105	0.07%	0.042 ($\sigma^2 = 0.015$)	0.906	0.0
revert edits	3,425	2.34%	0.040 ($\sigma^2 = 0.007$)	0.985	0.0
protect item	33	0.02%	0.032 ($\sigma^2 = 0.001$)	0.096	0.0
set description	6,692	4.57%	0.028 ($\sigma^2 = 0.002$)	0.961	0.0
add reference	10,960	7.48%	0.026 ($\sigma^2 = 0.004$)	0.925	0.0

Table 4.4: **Top 10 powerful edit types** ($n = 23839$, the list is sorted by the average JS distance value in a descending order).

The edit focus *term* involves simultaneous edits regarding *label*, *alias* and *description*, which makes its influence upon quality variation most significant ($avg = 0.043$), followed by reference ($avg = 0.041$) and edits ($avg = 0.040$), with their weights exhibited again as discussed above. It is worth noting that single activity regarding *claim* and *sitelink* contribute less then expected to the quality dynamics in spite of their huge amount being edited (56.82%). Similarly, Appendix 5.3 investigates the statistical behaviour with an activity type perspective. The most significant activity type *merge* ($avg = 0.055$) is performed when two or more items exist in Wikidata on the same object[9]. It is achieved by first pooling the collective data of various items together in a so-called recipient item, then conducting a redirection of the obsolete page to recipient item[10]. As this multi-step process normally requires consecutive activities as *delete* and *redirect*, it could be immensely influential according to the amount of contents deleted.

In addition, the last two columns of both tables demonstrate again the correlation of quality variation with article's current development stage. Although most of the peaks occurred at the stage D or E, activity *protect* reached its ceiling value at stage A, which can be explained by its edit nature that protecting a highly completed page is more valuable[11].

[9] Please note that the JS value distribution of merge is relative discrete ($\sigma^2 = 0.024$) compared to others, more details can be seen in Appendix 5.3

[10] `https://www.wikidata.org/wiki/Help:Merge` [Accessed: 06-Aug-2019]

[11] Users belonging to a specific user-group such as admins can protect pages for a short time to prevent vandalism or spam detected which repeatedly occurs on them. More information at `https://www.wikidata.org/wiki/Wikidata:Page_protection_policy` [Accessed: 06-Aug-2019]

ID	Sequence
1	<(set item) (set sitelink, set label) (add label)>
2	<(set sitelink) (set label) (set description)>

Length	Prefix	Projection DB	Occurrence
1	set item	<(set sitelink, set label) (add label)>	1
	set sitelink	<(_, set label) (add label)> <(set label) (set description)>	2
	set label	<(add label)> <(set description)>	2
2	set sitelink, set label	<(add label)> <(set description)>	2

Table 4.5: Example of prefix-based projection.

4.3 Sequence Mining

With a better understanding now of the dataset nature, given the observations made in preceding chapters, quantitative investigations employing sequential pattern mining are conducted in order to discover the actual frequent patterns responsible for the regularities in the event logs. As introduced before, there exist a variety of algorithms mining the frequent sequential patterns in time-related data (cf. Chapter 2.3), from which the PrefixSpan (prefix-projected sequential pattern mining) algorithm is chosen for mining our dataset. As a representative of pattern growth algorithms, PrefixSpan first scans the sequence databases and denotes the number of occurrences for all prefixes with length 1, where the prefixes as well as suffixes, i.e., the remainder of sequences are stored in a so-called projected database. It then uses the most frequently occurred sequence patterns as the prefix for the next iteration, and the whole process will continue as the prefix length grows until a threshold (minimum support) is reached (cf. Table 4.5). Compared with its opponent, Apriori-like algorithms, the PrefixSpan is selected based on the following properties (cf. [19]):

- **Avoid the huge set of candidate sequences**. Apriori-based algorithms operate in a join-prune fashion, where candidate sequences of all possible lengths are required to be generated. This means that the set of candidate sequences includes all permutations of activities that took place in an article, which could be huge even for a moderate seed set. According to Pei et al. [19], if there are $1,000$ frequent sequences of length 1, an Apriori-based algorithm will generate $1000 \times 1000 + \frac{1000 \times 999}{2} = 1,499,500$ candidate sequences. This time-consuming phase is retrenched in PrefixSpan as no candidate sequence needs to be generated as it grows longer sequential patterns from the shorter frequent ones.
- **Reduce the scan volume of databases**. An Apriori-based algorithm grows the length of each candidate sequence by one at each database scan, to find a sequential pattern of length 10, and the whole database must be scanned at least 10 times. In contrast to this, the projected databases generated in PrefixSpan algorithm keep shrinking as the postfix sub-sequences are the only ones to be further projected into a projection database, which significantly reduces the database volume to be scanned.
- **Superiority of mining long sequential patterns**. According to Pei et al. [19], the number of candidate sequences in Apriori-based algorithms is exponential to the length of sequential

patterns to be mined. For a length 10 sequential pattern, the number of candidate sequences to be generated is $\sum_{i=1}^{10} \binom{10}{i} = 2^{10} - 1 \approx 10^3$, which again, will be avoided by means of PrefixSpan as there will be no candidate sequence needed.

A pseudo-code representing the procedure for mining frequent sequential patterns with a data quality perspective is presented in Algorithm 1. The entire process is mainly composed of two parts: Class *SequenceGenerator* converts the raw dataset inputted into a sequence database, and insufficient edits, i.e., those not satisfying the JS-distance constraint, are pruned in this phase; Class *PrefixSpan* mines the sequence database converted by means of the PrefixSpan algorithm (cf. [19]), and frequency constraint is set-up with user-specified minimum support, along with the requirements of mining closed/maximal frequent patterns being taken into consideration in this phase (cf. Chapter 2.3).

Algorithm 1 Mining sequential patterns of Wikidata editing activities.

Input:

- Original csv dataset D
- List of sequence identification L
- Minimum frequency support *min_ support*
- Minimum data variation threshold *js_ threshold*
- (optional) Mine closed frequent patterns *closed=True*
- (optional) Mine maximal frequent patterns *maximal=True*

Output: The complete set of sequential patterns
Method:

1. Call **SequenceGenerator**(D, L, *js_ threshold*):
   a) Create new dataset D' by selecting the respective columns according to L.
   b) Scan D' once, prune the rows not satisfying the *js_ threshold*.
   c) Collapse the consecutive edits e committed by the same editor into one event α then aggregate each article's edits into one sequence $S_i =< \alpha_1, \alpha_2 ... \alpha_m >$.
   d) Create sequence database $S = [S_1, S_2 ... S_n]$.
2. Call **PrefixSpan**(S, *min_ support*, [optional]*closed=True*, [optional]*maximal=True*):
   a) Set frequent pattern $\alpha =<>$ with length $l = 0$ as initialization.
   b) For each frequent pattern α, create a α-projected database $S|\alpha$.
   c) Scan $S|\alpha$ once, for each frequent pattern α find the set of frequent events b such that
      i. b can be assembled to the last element of α to form a sequential pattern, or
      ii. $< b >$ can be appended to α to form a sequential pattern.
   d) Append each frequent event b to α to form a new frequent pattern α' with length $l + 1$.
   e) Construct α'-projected database $S|\alpha'$ for each α', repeat the same process with the new frequent pattern α' until no more frequent patterns can be discovered.
   f) Convert the patterns resulted into closed frequent patterns or maximal frequent patterns if necessary.

4.4 Behavioral Patterns

In this section, we analyze the pattern discovered between all article sequences, across different quality variation levels, and across different frequency constraints. Since one of the goals of this research is using sequences to identify the effectiveness in terms of data quality, it would be valuable to use the JS threshold to identify which patterns more or less affect the quality and use the frequency constraint to support this identification.

The patterns discovered are the results of sequential pattern mining, where each article is considered as a sequence containing all edit activities that have been carried out on this article. Analyzing these patterns enables us to study the editing sessions from a global perspective by discovering the common patterns reoccurring frequently across multiple articles. The length of patterns is an essential metric to evaluate the mining results, as more valuable sequential details are embedded in such consecutive editing sessions. Thus, both constraints of data quality variation and frequency support will be adjusted so that patterns with appropriate size and length are being discovered.

Edit Type					**Count**
1st	2nd	3rd	4th	5th	
add description	set description				268
update item	set claim				292
update item	set description				274
add reference	add description				266
add reference	add reference				273
set claim	set claim	add description			279
set claim	set claim	update item			274
set claim	set claim	add reference			272
set claim	add description	set claim			274
set claim	set label	set claim			272
set claim	set label	set description			267
set claim	add reference	set claim			276
set claim	add reference	set description			264
set description	set claim	set description			267
add reference	set claim	set claim			275
add reference	set claim	set description			265
set claim	set claim	set claim	set label		267
set claim	set claim	set description	set label		266
set claim	set description	set description	set description		281
set claim	set claim	set claim	set description	set description	271
set claim	set claim	set description	set description	set claim	265
set claim	set claim	set claim	set description	set claim	264

Table 4.6: **Most frequent edit type sequences with low article quality variation** ($n = 439$, js_threshold = 0.01, min_support = 0.6. The list is ordered first by sequence length then by edit focus alphabetically).

Table 4.6 presents the most frequent patterns with low-quality variations in lengths of 2-5 edit types. As can be seen from these results, a significant majority (82%) of patterns represent sequences with edit focus claim (set claim), sometimes followed or interrupted by editing sessions with edit

focuses term[12] or reference. This is not surprising, given that 36% of single edit activity focuses on the claim. A more interesting fact is the high ratio of claim and description coming in pairs (50%), as there is merely 8% of single edit activity focusing on description (cf. Table 5.2). Moreover, we notice that most of the frequent patterns in this phase (86%) are purely adding or setting new contents into articles.

Edit Type			Count
1st	2nd	3rd	
remove claim			55
set claim	set claim		134
set claim	add description		51
set claim	set description		54
set claim	update item		77
set claim	set label		54
set claim	add reference		115
set claim	set reference		42
update item	set claim		57
add reference	add description		47
add reference	update item		60
set reference	update item		60
add reference	add reference		97
add reference	set reference		56
set reference	add reference	set claim	46

Table 4.7: **Most frequent edit type sequences with middle article quality variation** (n = 418, js_threshold = 0.1, min_support = 0.1. The list is ordered first by sequence length then by edit focus alphabetically).

A more telling statistic in terms of data quality is presented by increasing the JS distance threshold from 0.01 to 0.1 as well as relaxing the frequency restriction from 0.6 to 0.1. Table 4.7 presents the most frequent patterns with middle-quality variations in lengths of 1-3 edit types. As Table 4.7 reveals, the sequential patterns with a higher quality ensurance are much shorter than those described in Table 4.6 with a lower one. The majority of patterns represent sequences with edit focus claim (60%), sometimes followed by editing sessions focusing on term, reference or item. Different from before, there are more patterns discovered in this phase focusing on reference (53%) and the activity types are becoming more diverse, given the occurrence of activity type remove (4 distinct activity types).

With modulating the JS distance threshold up to 0.5, the pattern discovered generated quite powerful article quality variations. Table 4.8 presents the most frequent patterns with high-quality variations in lengths of 1-2 edit types. As can be seen from the results, the identified patterns are becoming shorter once more, and instead of edit focus claim (42%), most of the sequences contain patterns focusing on reference (58%), and the activity types are becoming more diverse (5 distinct activity types) compared to those with moderate quality constraints. It is worth noting that the frequency restriction has to be adjusted to a very low level (0.01) to guarantee the appropriate pattern lengths, as highly frequent sequences leading to high quality variations are not able to be sustained for a long time.

[12]Edit focus term includes multiple edit focuses of label, alias or description.

Edit Type		Count
1st	2nd	
set label		11
remove claim	set claim	7
set claim	set claim	10
set claim	add reference	9
add description	add description	6
update item	update item	18
add reference	remove claim	6
add reference	set claim	6
set reference	set claim	18
set reference	add description	14
set reference	revert edits	8
set reference	update item	7

Table 4.8: **Most frequent edit type sequences with high article quality variation** ($n = 376$, js_threshold = 0.5, min_support = 0.01. The list is ordered first by sequence length then by edit focus alphabetically).

5 Conclusion

Event log data in large-scale collaborative ontology projects like Wikidata contain a variety of meta-data about who is contributing what, when and how. Investigating the sequential patterns in such event log data provides a richer description of the temporal dynamics underlying the online knowledge collaborations. This research adopts Keegan's [11] definition of event logs, specifies the process of identifying, schematizing, analyzing and interpreting sequences into the Wikidata domain, and, in addition, introduces data quality variation as a new event log feature, so that an event log can be encoded into a message not only about *who-what-when-how*, but also *how much*.

As a conceptual and methodological contribution, this research took ORES as a proxy to evaluate the temporal article quality and utilized the Jensen-Shannon distance as a metric for measuring the quality variations across event logs. To demonstrate the usability of the extended methodology, this research outlined through a case study a step-by-step procedure for employing sequence analysis in context of Wikidata. The case study illustrates the impact of sequential editing activities on data quality using a sequential pattern mining approach. The study collected and processed the meta-data of 500 randomly chosen Wikidata articles, classified the raw revision comments into highly-structured edit types, used the JS distance to capture the quality dynamics between adjacent edit types, and finally, identified frequent sequential editing patterns satisfying a specified quality constraint. With analysis of the impacts of different sequence identifications in terms of quality variations, this empirical case study can be seen as a pilot study for investigating sequential behaviour patterns with a focus of data quality. In the meantime, it provides some remarkable findings that deserve more discussion and elaboration in the future:

Quality measurement and ORES. The case study quantified the quality development by means of JS distance and this performs well. As the JS distance is bounded between 0 and 1, the whole dataset presents a wide variation range from 0 to 0.9876. However, with more than a half of all revisions holding JS distances of 0 (identical item quality distribution between adjacent revisions), the average quality variation is rather insignificant (0.0098). Moreover, this questioned the ability of ORES item-quality model to identify the subtle quality changes, since the contents involved exist in the model features.

The power of a single edit. Having conducted elementary statistical tests on the sample dataset, the study found that instead of an absolute value, the average power of a single edit type is rather context-dependent. The JS value of a particular edit type changes accordingly with edit focus, activity type and the current development stage of the article. Edits in regard to focuses *Term* and *Reference* triggered higher quality variations on average. Similarly, activity types such as *Merge* and *Revert* generate on average higher quality changes. The article quality variation triggered by a single edit appears variously susceptible according to different developing stages. Overall, the power of a single edit becomes increasingly weaker as the articles are being completed.

Frequent sequential patterns. Frequent sequential patterns between article sequences are discovered by adjusting the constraints of quality and frequency so that patterns with appropriate size and length are identified. In increasing the threshold of quality variation, the frequency constraint has to be strongly compromised to ensure consecutive edit type sequences. This indicates that

frequent sequences leading to high quality variations are not able to be sustained for a long time. Having investigated the identified patterns from both edit focus and activity type perspectives, as the quality variation threshold increases, editing activities related to *Reference* are occurring more frequently (from 32% to 58%), and the activity types are becoming more diverse.

In effect, in the future there is a necessity to cross test the identified results by scaling up the data collected. To investigate the effectiveness of different sequence identifications in terms of data quality, this research chose *Artifact* and *Activity* as the fundamental features of event logs, extended them with sub-features like *Name Space* and *Edit Type*, and tested the effectiveness regarding data quality by setting *Development Stage* as a variable. This can be seen as a call to extend the sequence identifications with *Performer* for examining the temporal participation patterns, or with *Order* to capture an overall development process, both with a data quality perspective. More importantly, the JS distance approach has a rich potential for measuring quality improvement in the context of ORES by setting a desired probability distribution as null hypothesis. Knowing the sequential patterns that lead frequently to a higher article quality, a recommendation system could be constructed to complement the work of high-quality maintenance in online knowledge collaborations.

Literature

[1] Rakesh Agrawal and Ramakrishnan Srikant. "Mining Sequential Patterns". In: *Proceedings of the Eleventh International Conference on Data Engineering*. ICDE '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 3–14. ISBN: 0-8186-6910-1. URL: http://dl.acm.org/citation.cfm?id=645480.655281.

[2] To Tu Cuong and Claudia Müller-Birn. "Applicability of Sequence Analysis Methods in Analyzing Peer-Production Systems: A Case Study in Wikidata". In: *Social Informatics - 8th International Conference, SocInfo 2016, Bellevue, WA, USA, November 11-14, 2016, Proceedings, Part II*. 2016, pp. 142–156. DOI: 10.1007/978-3-319-47874-6_11. URL: https://doi.org/10.1007/978-3-319-47874-6%5C_11.

[3] Fredo Erxleben et al. "Introducing Wikidata to the Linked Data Web". In: *The Semantic Web – ISWC 2014*. Ed. by Peter Mika et al. Cham: Springer International Publishing, 2014, pp. 50–65. ISBN: 978-3-319-11964-9.

[4] Samer Faraj, Sirkka L. Jarvenpaa, and Ann Majchrzak. "Knowledge Collaboration in Online Communities". In: *Organization Science* 22.5 (Sept. 2011), pp. 1224–1239. ISSN: 1526-5455. DOI: 10.1287/orsc.1100.0614. URL: https://doi.org/10.1287/orsc.1100.0614.

[5] Michael Färber et al. "Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO". In: *Semantic Web Journal* (2017), pp. 1–53.

[6] Minos N. Garofalakis, Rajeev Rastogi, and Kyuseok Shim. "SPIRIT: Sequential Pattern Mining with Regular Expression Constraints". In: *Proceedings of the 25th International Conference on Very Large Data Bases*. VLDB '99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 223–234. ISBN: 1-55860-615-7. URL: http://dl.acm.org/citation.cfm?id=645925.671514.

[7] Goldberger, Gordon, and Greenspan. "An efficient image similarity measure based on approximations of KL-divergence between two gaussian mixtures". In: *Proceedings Ninth IEEE International Conference on Computer Vision*. Oct. 2003, 487–493 vol.1. DOI: 10.1109/ICCV.2003.1238387.

[8] Jiawei Han, Jian Pei, and Yiwen Yin. "Mining Frequent Patterns Without Candidate Generation". In: *SIGMOD Rec.* 29.2 (May 2000), pp. 1–12. ISSN: 0163-5808. DOI: 10.1145/335191.335372. URL: http://doi.acm.org/10.1145/335191.335372.

[9] Jiawei Han et al. "FreeSpan: Frequent Pattern-projected Sequential Pattern Mining". In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '00. Boston, Massachusetts, USA: ACM, 2000, pp. 355–359. ISBN: 1-58113-233-6. DOI: 10.1145/347090.347167. URL: http://doi.acm.org/10.1145/347090.347167.

[10] Jiawei Han et al. "Frequent pattern mining: current status and future directions". In: *Data Mining and Knowledge Discovery* 15.1 (Aug. 2007), pp. 55–86. ISSN: 1573-756X. DOI: 10.1007/s10618-006-0059-1. URL: https://doi.org/10.1007/s10618-006-0059-1.

[11] Brian C. Keegan, Shakked Lev, and Ofer Arazy. "Analyzing Organizational Routines in Online Knowledge Collaborations: A Case for Sequence Analysis in CSCW". In: *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. CSCW '16. San Francisco, California, USA: ACM, 2016, pp. 1065–1079. ISBN: 978-1-4503-3592-8. DOI: 10.1145/2818048.2819962. URL: http://doi.acm.org/10.1145/2818048.2819962.

[12] S. Kim et al. "Dynamic and Succinct Statistical Analysis of Neuroscience Data". In: *Proceedings of the IEEE* 102.5 (May 2014), pp. 683–698. ISSN: 0018-9219. DOI: 10.1109/JPROC.2014.2307888.

[13] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. ISSN: 00034851. URL: http://www.jstor.org/stable/2236703.

[14] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research Methods in Human-Computer Interaction*. Wiley Publishing, 2010. ISBN: 0470723378, 9780470723371.

[15] J. Lin. "Divergence measures based on the Shannon entropy". In: *IEEE Transactions on Information Theory* 37.1 (Jan. 1991), pp. 145–151. ISSN: 0018-9448. DOI: 10.1109/18.61115.

[16] Florent Masseglia, Pascal Poncelet, and Maguelonne Teisseire. "Web Usage Mining: How to Efficiently Manage New Transactions and New Clients". In: *PKDD*. Vol. 1910. Lecture Notes in Computer Science. Springer, 2000, pp. 530–535.

[17] Carl H. Mooney and John F. Roddick. "Sequential Pattern Mining – Approaches and Algorithms". In: *ACM Comput. Surv.* 45.2 (Mar. 2013), 19:1–19:39. ISSN: 0360-0300. DOI: 10.1145/2431211.2431218. URL: http://doi.acm.org/10.1145/2431211.2431218.

[18] Claudia Müller-Birn et al. "Peer-production System or Collaborative Ontology Engineering Effort: What is Wikidata?" In: *Proceedings of the 11th International Symposium on Open Collaboration*. OpenSym '15. San Francisco, California: ACM, 2015, 20:1–20:10. ISBN: 978-1-4503-3666-6. DOI: 10.1145/2788993.2789836. URL: http://doi.acm.org/10.1145/2788993.2789836.

[19] Jian Pei et al. "PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth". In: *Proceedings of the 17th International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 215–224. ISBN: 0-7695-1001-9. URL: http://dl.acm.org/citation.cfm?id=645484.656379.

[20] Lev Ratinov and Dan Roth. "Design Challenges and Misconceptions in Named Entity Recognition". In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. CoNLL '09. Boulder, Colorado: Association for Computational Linguistics, 2009, pp. 147–155. ISBN: 978-1-932432-29-9. URL: http://dl.acm.org/citation.cfm?id=1596374.1596399.

[21] Amir Sarabadani, Aaron Halfaker, and Dario Taraborelli. "Building Automated Vandalism Detection Tools for Wikidata". In: *Proceedings of the 26th International Conference on World Wide Web Companion*. WWW '17 Companion. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 1647–1654. ISBN: 978-1-4503-4914-7. DOI: 10.1145/3041021.3053366. URL: https://doi.org/10.1145/3041021.3053366.

[22] Ramakrishnan Srikant and Rakesh Agrawal. "Mining sequential patterns: Generalizations and performance improvements". In: *Advances in Database Technology — EDBT '96*. Ed. by Peter Apers, Mokrane Bouzeghoub, and Georges Gardarin. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 1–17. ISBN: 978-3-540-49943-5.

[23] Steve Stemler. "An overview of content analysis". In: *Practical Assessment, Research & Evaluation* 7.17 (2001). ISSN: 1531-7714. URL: http://pareonline.net/getvn.asp?v=7&n=17.

[24] Simon Walk, Philipp Singer, and Markus Strohmaier. "Sequential Action Patterns in Collaborative Ontology-Engineering Projects: A Case-Study in the Biomedical Domain". In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. 2014, pp. 1349–1358. DOI: 10.1145/2661829.2662049. URL: https://doi.org/10.1145/2661829.2662049.

[25] Mohammed J. Zaki. "SPADE: An Efficient Algorithm for Mining Frequent Sequences". In: *Machine Learning* 42.1 (Jan. 2001), pp. 31–60. ISSN: 1573-0565. DOI: 10.1023/A:1007652502315. URL: https://doi.org/10.1023/A:1007652502315.

Appendix

5.1 Activity Coding Scheme

Edit Focus	Edit Type	Edit Summary	Paraphrase
Alias	set alias	wbsetaliases-set	Set a new alias list for a Wikibase entity, possibly overwrite
	add alias	wbsetaliases-add	Add new aliases to the existing alias list of a Wikibase entity, won't overwrite
	update alias	wbsetaliases-add-remove	Add new aliases to the alias list as well as remove aliases from the list
		wbsetaliases-update	Update the alias list of a Wikibase entity
Claim	set claim	wbsetclaim-create	Create an entire statement or claim
		wbcreateclaim	Create Wikibase claims
	add claim	wbcreateclaim-create	Create Wikibase claims
		wbsetclaimvalue	Set the value of a Wikibase claim
	update claim	wbsetclaim-update	Update an entire statement or claim
		wbremoveclaims-update	Update Wikibase claims by removing claim value
		wbsetclaim-update-qualifiers	Update the value of an existing qualifier
		wbsetclaim-update-rank	Update the value of an existing rank
	remove claim	wbremoveclaims	Remove Wikibase claims
		wbremoveclaims-remove	Remove Wikibase claims
Description	set description	wbsetdescription-set	Set a description for a single Wikibase entity, possibly overwrite
	add description	wbsetdescription-add	Set a description for a single Wikibase entity, won't overwrite
	remove description	wbsetdescription-remove	Set a new description for a single Wikibase entity, remove the old one

Edits	revert edits	undo	Undo the edits made by the most recent editor of any item or page which are potential vandalisms
		reverted edits	Revert the edits made by the most recent editor of any item or page which are potential vandalisms
		restored edits	Restore the edits made by the most recent editor of any item or page which are potential vandalisms
	unrevert edits	unrevert edits	Set the potentially wrongly reverted edits back
Item	set item	created page	Create a new page as new Wikibase entity
		wbsetentity	Create a new Wikibase entity, possibly overwrite
		wbeditentity-create	Create a single new Wikibase entity, possibly overwrite
		wbeditentity-override	Create a new Wikibase entity, possibly overwrite
	add item	wbeditentity	Add a single new Wikibase entity and modifies it with serialized information
	update item	wbeditentity-update	Update existing Wikibase entity, possibly overwrite
	merge item	wbmergeitems	Merge multiple items
	redirect item	wbcreateredirect	Create entity redirects
	protect item	protected item	Protect an item page
	unprotect item	remove protection	Unprotect an item page
Label	set label	wbsetlabel-set	Set a label for a single Wikibase entity, possibly overwrite
	add label	wbsetlabel-add	Set a label for a single Wikibase entity, won't overwrite
	remove label	wbsetlabel-remove	Set a new label for a single Wikibase entity, remove the old one
Qualifier	add qualifier	wbsetqualifier-add	Create a qualifier or sets the value of an existing one to the claim, won't overwrite
	update qualifier	wbsetqualifier-update	Update the value of existing qualifier

	remove qualifier	wbremovequalifiers-remove	Remove one or more qualifiers from the claim
Reference	set reference	wbsetreference	Create a reference or set the value of an existing one
		wbsetreference-set	Set a new reference
	add reference	wbsetreference-add	Add value of existing reference
	remove reference	wbremovereferences	Remove one or more references of the same statement
		wbremovereferences-remove	Remove one or more references of the same statement
Sitelink	set sitelink	wbsetsitelink-set	Associate a page on a wiki with a Wikibase item
		wbsetsitelink-set-badges	Associate a page on a wiki with a Wikibase item, set badges
		wbsetsitelink-set-both	Associate a page on a wiki with a Wikibase item, set both link-title and badges
		wblinktitles-connect	Associate two pages on two different Wikis with a Wikibase item
	add sitelink	wbsetsitelink-add	Add a sitelink to the item, if the sitelink does not exist
		wbsetsitelink-add-both	Add a sitelink to the item, if the sitelink does not exist, add both link-title and badges
	update sitelink	clientsitelink-update	Update the existing link to a page from an item
	remove sitelink	wbsetsitelink-remove	Change the link to a page from an item, remove the link
		clientsitelink-remove	Change the link to a page from an item, remove the link (old version)
Term	set term	wbsetlabeldescriptionaliases	Set a new label, alias and description for a single Wikibase entity
	remove term	remove terms	Remove labels, aliases, and descriptions from an existing entity

5.2 Statistics of JS-distance with an Edit Focus Perspective

Focus	Count	%	Max./Min.	Average	Dev. Stage	Average
Term	315	0.21%	0.879/0.0	0.043 ($\sigma^2 = 0.006$)	C	0.074
					B	0.112
					A	0.009
Reference	12,757	8.71%	0.925/0.0	0.041 ($\sigma^2 = 0.011$)	E	0.192
					D	0.230
					C	0.071
					B	0.031
					A	0.009
Edits	3,426	2.33%	0.985/0.0	0.040 ($\sigma^2 = 0.007$)	E	0.081
					D	0.074
					C	0.046
					B	0.032
					A	0.029
Item	9,568	6.53%	0.988/0.0	0.024 ($\sigma^2 = 0.007$)	E	0.036
					D	0.043
					C	0.019
					B	0.033
					A	0.014
Description	11,879	8.11%	0.952/0.0	0.023 ($\sigma^2 = 0.007$)	E	0.066
					D	0.157
					C	0.018
					B	0.020
					A	0.021
Label	7,915	5.41%	0.937/0.0	0.021 ($\sigma^2 = 0.002$)	E	0.024
					D	0.028
					C	0.016
					B	0.018
					A	0.026
Claim	52,343	35.75%	0.941/0.0	0.020 ($\sigma^2 = 0.003$)	E	0.089
					D	0.103
					C	0.017
					B	0.013
					A	0.008
Alias	7,133	4.87%	0.873/0.0	0.010 ($\sigma^2 = 0.001$)	E	0.010
					D	0.019
					C	0.011
					B	0.008
					A	0.007
					E	0.011
					D	0.020

Qualifier	10,270	7.01%	0.161/0.0	0.006 ($\sigma^2 = 0.001$)	C	0.006
					B	0.007
					A	0.005
Sitelink	30,844	21.07%	0.898/0.0	0.004 ($\sigma^2 = 0.001$)	E	0.002
					D	0.014
					C	0.009
					B	0.010
					A	0.007

Table 5.2: **Statistics of JS distance value with an edit focus perspective** ($n = 146450$, focus *term* is a combination of multiple focuses *label*, *alias* and *description*, and focus *edits* refers to previous edits detected as potential vandalism edits, often appears with activity *revert*. The list is sorted by the average JS-distance value in a descending order).

5.3 Statistics of JS-distance with an Activity Type Perspective

Activity	Count	%	Max./Min.	Average	Dev. Stage	Average
Merge	512	0.35%	0.988/0.0	0.055 ($\sigma^2 = 0.024$)	E	0.261
					D	0.008
					C	0.024
					B	0.052
					A	0.027
Revert	3,425	2.34%	0.985/0.0	0.040 ($\sigma^2 = 0.007$)	E	0.081
					D	0.061
					C	0.046
					B	0.032
					A	0.029
Remove	8,133	5.55%	0.932/0.0	0.022 ($\sigma^2 = 0.004$)	E	0.113
					D	0.070
					C	0.017
					B	0.018
					A	0.007
Set	70,862	48.39%	0.985/0.0	0.019 ($\sigma^2 = 0.004$)	E	0.037
					D	0.069
					C	0.021
					B	0.021
					A	0.009
Add	39,596	27.04%	0.952/0.0	0.019 ($\sigma^2 = 0.002$)	E	0.012
					D	0.079
					C	0.029
					B	0.013
					A	0.011
Update	23,838	16.28%	0.941/0.0	0.017 ($\sigma^2 = 0.002$)	E	0.017
					D	0.033
					C	0.017
					B	0.024
					A	0.010
Protect	33	0.02%	0.096/0.0	0.032 ($\sigma^2 = 0.001$)	C	0.0
					B	0.001
					A	0.048
Redirect	49	0.03%	0.001/0.0	0.001($\sigma^2 = 0.001$)	E	0.001

Table 5.3: **Statistics of JS distance value with an activity type perspective** ($n = 146448$, activity type *unprotect* and *unrevert* with a size of 1, i.e. standard deviation of 0 are excluded. The list is sorted by the average JS-distance value in a descending order).