University of Zurich^UZH

Bachelor's Thesis 16/03/2019

Modelling and Importing Dynamic Data into Wikibase

A Case Study of the Swiss Transportation System

Samuel Meuli
of Zurich, Switzerland

Student ID: 15-708-530
samuel.meuli@uzh.ch

Advisor: **Cristina Sarasua**
Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
https://www.ifi.uzh.ch/ddis

Acknowledgements

I would like to thank the Department of Informatics at the University of Zurich, the Dynamic and Distributed Information Systems Group (DDIS), and Prof. Abraham Bernstein for giving me the opportunity to write this bachelor's thesis. Furthermore, I would like to thank my advisor, Cristina Sarasua, for her guidance and feedback.

Abstract

The Swiss Federal Railways (SBB) publish datasets on their transportation network in the GTFS format. The company is now looking to integrate this information into the Wikidata ecosystem. The datasets are updated every week with possible changes to the network. The goal of this thesis is to provide users with a way to get an impression of the network's evolution over time. For this purpose, a software tool for mapping GTFS data to Wikibase entities as well as importing and updating these in an instance of Wikibase is developed. To make such graph dynamics understandable for humans and machines, an RDF ontology for modelling changes is defined and a statistical analysis of the SBB's datasets is performed.

Zusammenfassung

Die Schweizerischen Bundesbahnen (SBB) veröffentlichen Datensätze über ihr Transport-Netzwerk im GTFS-Format. Die Firma möchte diese Informationen nun ins Wikidata-Ökosystem integrieren. Die Datensätze werden jede Woche mit möglichen Änderungen im Netzwerk aktualisiert. Ziel dieser Arbeit ist es, Nutzern zu ermöglichen, sich einen Eindruck über die Entwicklung des Netzwerks über die Zeit zu verschaffen. Dazu wird ein Software-Tool entwickelt, welches das Modellieren von GTFS-Daten in Wikibase sowie deren Import und die Aktualisierung in einer Wikibase-Instanz erlaubt. Um die Dynamik solcher Graphen für Menschen und Computer verständlich zu machen, wird eine RDF-Ontologie für die Modellierung von Änderungen definiert und eine statistische Analyse der SBB-Datensätze durchgeführt.

Contents

List of Figures viii

List of Tables ix

List of Listings ix

List of Acronyms x

1 Introduction 1

1.1 Background 1

1.2 Thesis Scope 2

1.3 Outline 3

2 Foundations 4

2.1 The Semantic Web 4

2.1.1 Overview 4

2.1.2 RDF 4

2.1.3 Vocabularies 5

2.1.4 Syntaxes 5

2.2 Wikidata and Wikibase 6

2.2.1 The History of Wikidata 6

2.2.2 The Wikibase Data Model 7

2.2.3 Wikibase and RDF 8

2.3 GTFS 9

3 Modelling Time in RDF and Wikibase 10

3.1 Overview 10

3.2 Modelling Time in RDF 10

3.2.1 Versioning and Labelling 10

3.2.2 Valid and Transaction Time 11
3.3 Modelling Time in Wikibase 12

4 Data Mapping and Import 13
4.1 Overview 13
4.2 Data Mapping 14
4.2.1 Identifying Relevant Data in GTFS Feeds 14
4.2.2 Mapping GTFS Data to the Wikibase Data Model 17
4.3 Import of Static Data 20
4.3.1 Overview 20
4.3.2 Components 20
4.3.3 Deployment 25
4.4 Import of Dynamic Data 25
4.4.1 Updating Data in Wikibase 25
4.4.2 Punctuality Information 27

5 Representation of Dynamics 28
5.1 Update History and Summarisation 28
5.2 Dynamics of a Single Entity 28
5.3 Dynamics of a Dataset 29

6 Analysis 32
6.1 Overview 32
6.2 Analysis of Network Dynamics 32
6.3 Analysis of Punctuality 33
6.3.1 Survey Results 33
6.3.2 Punctuality over Time 34

7 Discussion 35
7.1 Results 35
7.1.1 Network Dynamics 35
7.1.2 Punctuality 36
7.1.3 Dynamics Vocabulary 36
7.2 Challenges 37
7.3 Limitations 37
7.4 Future Directions 39

8 Conclusions 40

A Appendix 41
A.1 GTFS–Wikibase Mapping . . . 41
A.1.1 Mapping tables . . . 41
A.1.2 Turtle Representation . . . 43
A.1.3 Representation in DCAT and VoID Vocabularies . . . 45
A.2 Extended GTFS Route Types . . . 47
A.3 Wikibase Items . . . 52
A.4 Dynamics Vocabulary . . . 53
A.4.1 Turtle Representation . . . 53
A.4.2 Wikibase Representation . . . 58
A.5 Survey . . . 60
A.5.1 Survey Questions . . . 60
A.5.2 Survey Results . . . 62
A.6 Software . . . 63
A.7 Data Analysis . . . 63

References 83

List of Figures

2.1 RDF triple 5
2.2 RDF example using vocabularies 6
2.3 Structure and example of a Wikibase statement 8

4.1 Component diagram 14
4.2 GTFS structure 16
4.3 GTFS–Wikibase mapping 18
4.4 Class diagram of `python-wikibase` (left half) 23
4.5 Class diagram of `python-wikibase` (right half) 24
4.6 Sequence diagram 26

5.1 Vocabulary for modelling dataset dynamics 31

A.1 Stop item in Wikibase 52
A.2 Stop item history in Wikibase 53
A.3 Dynamics vocabulary in Wikibase (first part) 58
A.4 Dynamics vocabulary in Wikibase (second part) 59
A.5 Punctuality survey (first part) 60
A.6 Punctuality survey (second part) 61
A.7 Results of punctuality survey 62

List of Tables

4.1 GTFS files 15

A.1 Mapping of `agency.txt` 41
A.2 Mapping of `routes.txt` 41
A.3 Mapping of `stops.txt` 42
A.4 Mapping of Extended GTFS Route Types to Wikibase items 48

List of Listings

4.1 `wikibase-api` usage example 21
4.2 `python-wikibase` usage example 22

A.1 GTFS–Wikibase mapping in Turtle 43
A.2 GTFS–Wikibase mapping in DCAT 45
A.3 GTFS–Wikibase mapping in VoID 47
A.4 Vocabulary for modelling dataset dynamics 53

List of Acronyms

API	application programming interface
CSV	comma-separated values
DCAT	Data Catalog Vocabulary
FOAF	friend of a friend
GTFS	General Transit Feed Specification
HTTP	Hypertext Transfer Protocol
IRI	Internationalised Resource Identifier
JSON-LD	JavaScript Object Notation for Linked Data
LOD	linked open data
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SBB	Schweizerische Bundesbahnen (Swiss Federal Railways)
SPARQL	SPARQL Protocol and RDF Query Language
Turtle	Terse RDF Triple Language
VoID	Vocabulary of Interlinked Datasets
XML	Extensible Markup Language

1 Introduction

1.1 Background

Every day, the Swiss Federal Railways (SBB) transport 1.26 million passengers in 10,671 trains between 793 stations (Meier & Osterwald, 2018). With such an expansive transportation network and high number of customers, the company generates a large amount of data.

Since 2016, the SBB have been making a significant part of their data available as open data[1]. Publishing such information increases the SBB's transparency and promotes innovation from other individuals and companies, which, in turn, is beneficial for the SBB[2]. As part of this Open Data initiative and in order to increase their data's discoverability, the company is looking to integrate a subset into the Wikimedia ecosystem. More specifically, the SBB are interested in publishing data about their network's topology using Wikidata's graph data store, which facilitates an analysis of their network's evolution over time.

The SBB's transportation network is dynamic, e.g. connections and schedules may change over time. The company is interested in exposing information on these network dynamics. Because, potentially, the degree to which the network changes may be high and because changes may be small and not of particular interest to the user, a way to summarise these changes needs to be found.

[1] *https://data.sbb.ch* since 2016 and *https://opentransportdata.swiss* since 2017

[2] See slides of related talk by C. Trachsel: *http://www.digitale-nachhaltigkeit.unibe.ch/unibe/portal/fak_wiso/a_bwl/inst_wi/abt_digital/content/e90958/e490158/e636040/e652610/e682224/OpenDataVorlesung2018_Termin08_Gastreferat_SBB_Trachsel_ger.pdf*

1.2 Thesis Scope

On a weekly basis, the SBB make a dataset with information about their network and current timetable available in the GTFS format[3]. The transport network's topology can be extracted from these datasets. One goal of this thesis is the development of software which maps this information to Wikibase entities and imports them into a custom, transport-specific Wikibase instance. This allows the consumption of the SBB's data in RDF, and, thanks to Wikibase's SPARQL endpoint, the execution of complex queries on the dataset, allowing users to analyse the data or use it for other applications. Furthermore, users can easily delve in the SBB's data using the familiar MediaWiki interface. In the future, this Wikibase instance could be linked to Wikidata or its contents could be imported directly. The information on Wikidata could then be used in Wikipedia articles.

To be able to gain a deeper understanding of the transportation network's evolution over time, the software tool needs be able to handle dynamic information. Changes in the GTFS dataset are detected and analysed, and the Wikibase instance is updated with these changes. Because GTFS is a widely used standard, the resulting software is compatible with any specification-conformant dataset. For example, it is possible to integrate feeds by other transport companies into the same transport Wikibase instance in the future.

Due to the potentially high amount and frequency of changes in datasets like the SBB's, analysing changes on the level of a single item does not inform the observer about the dynamics of the network. Because Wikibase does not provide a summarised view of the dynamics, such a representation of changes on the dataset level is developed in this thesis. For this purpose, an extension to the DCAT vocabulary is created which exposes such a summary in a graph-like fashion.

In a final step, a simple descriptive data analysis is performed on the information exposed using the DCAT extension in order to get an understanding of the SBB's network dynamics over time. Because the SBB also make information on punctuality publicly available, key numbers from these datasets are integrated into the mapped GTFS entities and analysed together with the network dynamics.

The goals of this thesis are the following:

1. Implementation of a software tool for mapping the SBB's GTFS data to Wikibase entities and importing/updating them in a custom Wikibase instance

[3] Open Data Platform Swiss Public Transport: *https://opentransportdata.swiss*

2. Development of an extension to the DCAT ontology for making information on performed updates available to humans and machines
3. Statistical analysis of the dynamics in the SBB's GTFS feeds and the punctuality of their public transport connections

1.3 Outline

Chapter 2 provides an overview of relevant concepts and tools. Chapter 3 discusses different approaches for modelling time in RDF, explains their application to Wikibase, and shows the need for a summarised representation of dynamics on the level of a dataset.

In section 4.2, the structure of a GTFS feed is explained and the mapping process from GTFS to Wikibase entities is illustrated. Section 4.3 describes the different layers of the mapping/import software and their interaction. The extensions to these software components required for the modelling of dynamic data as well as the integration of data on punctuality is described in section 4.4. A suitable representation of punctuality information is identified using a survey.

Based on the need for an ontology for dataset dynamics, the design of such a vocabulary is discussed in chapter 5 and its application to the SBB dataset is shown. In chapter 6, a descriptive data analysis is performed on this summarised data.

At the end of this thesis, its results, challenges, and limitations are discussed and possible future directions are outlined (chapter 7).

2 Foundations

2.1 The Semantic Web

2.1.1 Overview

The World Wide Web was designed so its content would be readable for people. While humans can understand the structure and meaning of different parts of a web page, this task is not as straightforward for a computer (Berners-Lee, 2000). The Semantic Web, a web of data with the purpose of additionally being processable by machines, is an attempt to solve this problem. It is considered an extension of the existing web, something it will eventually evolve into (Berners-Lee, Hendler & Lassila, 2001).

This web of linked data is created by structuring data in a way that makes it understandable for machines and that allows the specification of relations between different entities. Using such structured data and a set of inference rules, information can be queried more easily (e.g. using the SPARQL Protocol and RDF Query Language) and machines are able to understand relationships and perform automated reasoning. (Berners-Lee et al., 2001; W3C SPARQL Working Group, 2013)

2.1.2 RDF

The Resource Description Framework (RDF) is a graph-based framework for representing linked data. An RDF graph is of a set of triples, each of which comprises a subject, a predicate, and an object. Subject and object correspond to the graph's vertices, predicates to its edges (see figure 2.1). Using such triples, arbitrary statements can be created. Subject, predicate, and object may be of the following types (Cyganiak et al., 2014):

- **Subject:** IRI or blank node

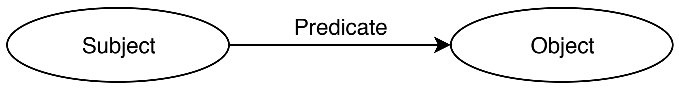


Figure 2.1: An RDF graph consisting of a single triple with subject, predicate, and object. Based on Cyganiak et al. (2014)

- **Predicate:** IRI
- **Object:** IRI, literal, or blank node

An Internationalised Resource Identifier (IRI) is a Uniform Resource Identifier (URI) with a larger set of allowed characters. It uniquely identifies a resource. IRIs allow to link different resources and are therefore fundamental for the Semantic Web. A literal is a value with a data type and optionally a language tag. (Cyganiak et al., 2014)

2.1.3 Vocabularies

An RDF vocabulary defines a set of IRIs which can be used in RDF graphs (Cyganiak et al., 2014). Such vocabularies, also known as ontologies, can be used to bring structure and meaning to the represented information (Thalhammer, 2017).

The Resource Description Framework Schema (RDFS) vocabulary extends RDF by additional semantic constraints, e.g. by specifying `rdfs:Class` and `rdfs:Datatype` (Hayes & Patel-Schneider, 2014). Using RDFS, a vertex *Professor* could be defined as a subclass (`rdfs:subClassOf`) of *University employee*. If *John Doe* works as a *Professor*, software can then infer that he also counts as a *University employee*.

Other vocabularies, which are often defined for a specific area or topic, can build on these foundations. For instance, the FOAF vocabulary defines terms on topics such as people, human collaboration, and association (Brickley & Miller, 2014). Figure 2.2 shows an example using FOAF.

2.1.4 Syntaxes

Various serialisation formats for RDF exist. Popular syntaxes include RDF/XML[1], Turtle[2], and JSON-LD[3]. For listings in this work, Turtle is used.

[1] RDF 1.1 XML Syntax: *https://www.w3.org/TR/rdf-syntax-grammar*
[2] RDF 1.1 Turtle: *https://www.w3.org/TR/turtle*
[3] JSON-LD 1.1: *https://w3c.github.io/json-ld-syntax*

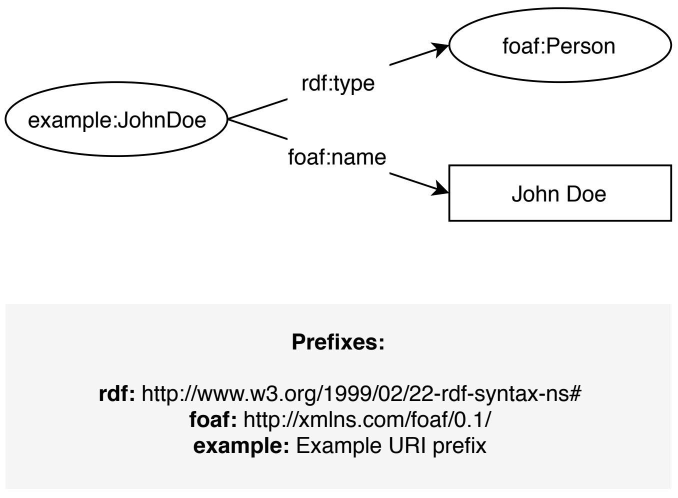

example:JohnDoe
rdf:type
foaf:Person
foaf:name
John Doe
Prefixes:
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
foaf: http://xmlns.com/foaf/0.1/
example: Example URI prefix

Figure 2.2: RDF graph with triples using the RDF namespace and FOAF vocabulary

2.2 Wikidata and Wikibase

2.2.1 The History of Wikidata

Wikipedia launched on January 15, 2001 and quickly became the largest reference website in the world (Ayers, Matthews & Yates, 2008). As of December 2018, Wikipedia exists in 293 languages, with the English version alone containing nearly 5.8 million articles ('List of Wikipedias', 2018).

With Wikipedia's content growing rapidly, some problems regarding the management of facts became clear. For instance, certain language-independent information like the age of a person or the population of a country would need to be manually updated in each language version of an article. Besides requiring a lot of manual effort, this could result in inconsistencies between versions. (Vrandečić, 2013)

Wikidata, which launched in October of 2012, was created to solve problems like the aforementioned one. It is a knowledge base for structured data, which – like Wikipedia – can be accessed and edited by anyone. Wikidata provides useful features such as handling of multilingual information, management of data sources, a SPARQL query interface, and the ability to make edits in bulk using bots. Wikidata has also seen rapid growth and its data is widely used. Besides serving as a source of data for Wikipedia, e.g. for statistics in an article's infobox across all languages, it is also used by Google's Knowledge Graph, Facebook's Open Graph, and Wolfram Alpha. (Krötzsch & Vrandečić, 2014)

2.2.2 The Wikibase Data Model

Wikidata is powered by a number of extensions to MediaWiki, the wiki software which is also used by Wikipedia. The main part of the tool, which implements the data model used by the Wikidata project, is called Wikibase. It is open source software and can be installed and used by anybody for free[4]. (Vrandečić, 2013)

What follows is an overview of Wikibase's data model ('Wikibase/DataModel', 2018). The Wikidata item *Switzerland*[5] is used as an example.

In Wikibase, 'things', for example the person *Douglas Adams* or the country *Switzerland*, are represented by **items**. Relationships between items are described using **properties**. Items and properties are both **entities** and identified by an item or property URI (e.g. 'Q39' for *Switzerland*). Items have the URI prefix 'Q'; properties have the prefix 'P'. Entities can be described using information of different types:

- The entity's name is called **label**. An entity may have a different label for each language. For instance, the English label of *Switzerland* is 'Switzerland', whereas the German label is 'Schweiz'.

- **Aliases** are alternative labels for entities and are also defined per language. For example, *Switzerland* is also known as 'Swiss Confederation' in English.

- The **description** provides a brief explanation of the meaning of the entity, e.g. 'federal republic in Western Europe' is the English description for *Switzerland.* An entity's combination of label and description must be unique within each language.

- **Statements** are used to provide information about an entity. A statement for an entity consists of the following elements (see figure 2.3):
  - A property, which is used to describe the relationship between the item and the statement's value.
  - A **value**, which may be a reference to another entity or a value with a **data type** (e.g. a string, number, date, or geographic location). Alternatively, the statement may have no value or a value that is specified to be unknown.
  - A list of **qualifiers**, which are used to provide further details about the statement. For instance, a start and end date between which a value was valid could be specified. Like statements, qualifiers consist of a property and a value.

[4] Overview of Wikibase components and links to their source code: *http://wikiba.se*
[5] *Switzerland* on Wikidata: *https://www.wikidata.org/wiki/Q39*

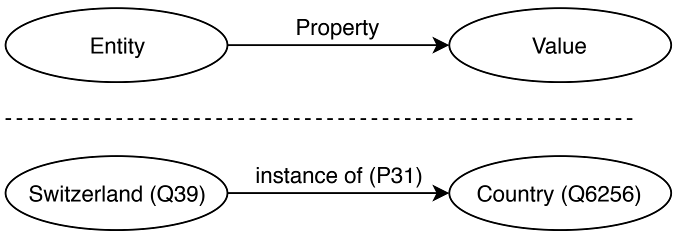


Figure 2.3: Structure and example of a Wikibase statement (without qualifiers and references)

  - A list of **references**, which can be used to indicate the source(s) of the statement. This might be the URL to another database. References also consist of a property and a value.
  - A **rank** – either 'preferred', 'normal', or 'deprecated' – for highlighting the importance, recency, or reliability of a statement. Ranks also affect how statements are displayed in the MediaWiki interface.

  Property, value, and the corresponding qualifiers together form a **claim**.

- **Site links** are links from Wikibase entities to other WikiMedia sites.

2.2.3 Wikibase and RDF

Wikibase's data model is similar to RDF and therefore makes its data available in the RDF format, both as a dump of the entire database and as linked open data (LOD) endpoints for individual entities. The LOD endpoints can be reached by using the corresponding `Accept` HTTP request header or by appending the file extension of the desired serialisation format to the entity URL (e.g. *https://wikidata.org/entity/Q1.ttl* for fetching its Turtle representation). (Erxleben, Günther, Krötzsch, Mendez & Vrandečić, 2014; 'Wikidata:Data Access', 2019)

However, the existence of qualifiers and references makes Wikibase's data model more complex and therefore more difficult to represent in RDF. Different solutions are possible for dealing with this problem, e.g. the application of reification, 'the process of encoding complex structures in RDF by introducing new individuals to represent them' (Erxleben et al., 2014). But even after reifying the statements, the resulting RDF graph is relatively complex. For figures and listings in this thesis, qualifiers and references are therefore omitted to make them easier to understand.

2.3 GTFS

The General Transit Feed Specification (GTFS) is a standard format for public transportation data defined by Google. A GTFS transit feed contains information about stops, connections, schedules, etc. The specification describes a GTFS feed as a ZIP file containing a collection of TXT files formatted as comma-separated values (CSV). The files contained in a GTFS feed are explained in more detail in section 4.2. (‘GTFS Static Overview’, 2016)

The GTFS format has been widely adopted by transport companies, partly because Google has been offering the integration of such data into its Google Transit and Google Maps services. This has led to a large number of transportation datasets being made available as open data. (Antrim & Barbeau, 2013; ‘GTFS Static Overview’, 2016)

3

Modelling Time in RDF and Wikibase

3.1 Overview

To be able to represent dynamics in graphs, a time dimension needs to be introduced. A significant amount of research has been conducted on this topic. The approaches can be coarsely grouped into methods which are based on versioning and methods which make use of labelling. These two approaches are described in section 3.2.

Section 3.3 explains how time is represented in Wikibase and how the aforementioned concepts apply. It is discussed what can be done to provide more information about the updates performed at different time points in Wikibase.

3.2 Modelling Time in RDF

3.2.1 Versioning and Labelling

Modelling time in RDF requires providing a means to represent, query, and update time information for a graph. Generally, the differentiation between two general concepts for adding a time dimension to non-temporal graphs has been made in the literature: Labelling and versioning. These two concepts are presented and discussed in what follows. (Gutierrez, Hurtado & Vaisman, 2007)

Versioning. A time dimension can be added to a graph by storing a snapshot of the entire graph for each point in time at which an update occurs. This versioning process is also known as the snapshot model. Versioning has the advantage that queries for data at a specific point in time are identical to queries in a nontemporal graph due to the data model remaining the same. Queries are therefore equally simple to formulate. (Gutierrez et al., 2007; Tappolet & Bernstein, 2009)

The snapshot model, however, brings about some considerable disadvantages. If small changes are made to the dataset at many different points in time, significant overhead is incurred because a snapshot needs to be taken of the entire graph and not just of the affected triples. Furthermore, time is only exposed implicitly through snapshots, which makes the understanding of the data and the formulation of queries more difficult. (Tappolet & Bernstein, 2009)

One way the snapshot model can be implemented is by using named graphs. An additional graph can be created which holds the temporal information that links the snapshot graphs to specific points in time[1].

Labelling. Labelling is based on storing temporal information as part of the data model. It is also known as the timestamp model. Using this approach, the overhead from storing the entire graph for every point in time at which changes are made is avoided. Therefore, labelling requires significantly less storage than versioning. Furthermore, the model provides a simple way to make queries of time points at which certain conditions hold. (Gutierrez et al., 2007)

For the implementation of the timestamp model, a vocabulary like *Temporal RDF* (Gutierrez et al., 2007) could be used to ensure compatibility with the RDF syntax. This ontology can be used to define the time of validity for triples in a graph. A disadvantage of this labelling approach is that whenever time information should be added to a relationship, techniques like reification need to be applied (Tappolet & Bernstein, 2009).

3.2.2 Valid and Transaction Time

Two important concepts related to time representation are valid and transaction time. Valid time is the time during which a fact holds in the world. Transaction time is the time during which a fact is represented in the database. The snapshot model represents transaction time; the timestamp model is normally used for modelling valid time. (Gutierrez et al., 2007)

[1] This implementation approach is discussed in a blog article by I. Davis: *http://blog.iandavis.com/2009/08/representing-time-in-rdf-part-2*

3.3 Modelling Time in Wikibase

For modelling the dynamics of a dataset in Wikibase, two views on the data are of interest: The view of the changes to a single entity and the view of the updates to the entire dataset.

Wikibase stores the history of changes to a single entity and makes the data available in the MediaWiki interface, where users can browse the item's version history. This is an implementation of the timestamp model and the represented time information corresponds to transaction time. Information about valid time can be added as qualifiers for claims, e.g. with the *start date* and *end date* properties.

On the level of the entire dataset, Wikibase does not provide users with a way to analyse changes. Because statistics like the number of changes made at a certain point in time might still be of interest, an attempt to address this problem is made in this thesis. In datasets like the SBB's GTFS feeds, changes to the individual rows are not of particular importance for their representation in Wikibase because they may undergo small changes with high frequency. In order to be able to draw conclusions about the dynamics of the transport network, it is important to have a view on the changes from further away. The goal is to find an adequate way to summarise the changes to the database during an update in a way that presents the relevant information to the consumer in an understandable way. The suggested approach corresponds to the snapshot approach.

It is common that linked open datasets are described using dedicated ontologies like Data Catalog Vocabulary (DCAT) and Vocabulary of Interlinked Datasets (VoID). Therefore, it makes sense to extend such a vocabulary with the aforementioned change information on the dataset level. This approach makes it simple to access the statistics with existing tools, makes them readable for machines, and allows extending the resulting graph structure with other information if required. DCAT, VoID, and the proposed vocabulary extension are described in more detail in chapter 5.

4 Data Mapping and Import

4.1 Overview

The main contribution of this thesis is the software for modelling and importing entities from a GTFS feed into Wikibase. The modelling process comprises identifying the parts of a GTFS feed that should be imported into Wikibase and finding an adequate mapping of these items to Wikibase entities. The developed software solution takes a GTFS dataset as input, applies the defined mapping algorithm, and imports the data into an instance of Wikibase. The software is built in a modular fashion so parts of the code can be reused in other software. For instance, the GTFS–Wikibase mapping script is not only applicable to the SBB's dataset but to feeds of any transport company which implement the GTFS specification. The software is composed of the following modules. They are listed in a bottom-up fashion, from general to application-specific:

- `wikibase-api`: A wrapper around the Wikibase application programming interface (API) for simpler authentication and API requests
- `python-wikibase`: An object-oriented abstraction of the Wikibase API
- `gtfs-to-wikibase`: A module which parses a GTFS feed and maps and imports the contained entities into an instance of Wikibase
- `sbb-wikibase`: A tool for downloading the latest SBB dataset and importing it into Wikibase[1]

[1] Besides the functionality mentioned above, this module also contains other code related to punctuality and data analysis (described later in this thesis).

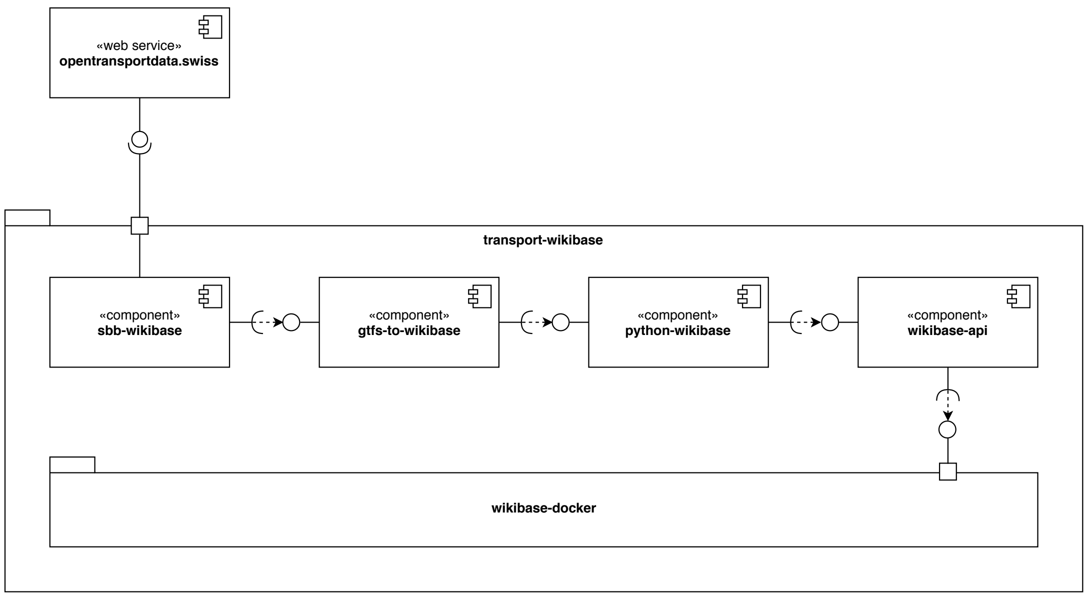


Figure 4.1: UML component diagram of the four libraries

A component diagram of the libraries' interaction can be seen in figure 4.1.

The developed solution – specifically `gtfs-to-wikibase` – is built so the tool can detect changes in more recent versions of the GTFS dataset and apply the updated entity mapping in Wikibase. Users can inspect the changes using the version history integrated into the MediaWiki interface. Furthermore, to be able to view or change information on the dataset level, a module for generating entities of the proposed extension to the DCAT vocabulary is integrated into the `sbb-wikibase` module.

The aforementioned entity mapping and software tools are described in more detail in the following sections.

4.2 Data Mapping

4.2.1 Identifying Relevant Data in GTFS Feeds

A GTFS feed is a collection of text files structured in the CSV format. The GTFS Static specification defines 13 different such files, some of which are optional. An overview of the files and their contents can be found in figure 4.1. ('GTFS Static Overview', 2016)

Table 4.1: GTFS files and their contents ('General Transit Feed Specification Reference', 2018)

File name	**Required**	**Content**
`agency.txt`	Yes	Transport agencies
`stops.txt`	Yes	Served stations
`routes.txt`	Yes	Transport routes (e.g. connections between stop *a* and *b*)
`trips.txt`	Yes	Instances of transport routes (e.g. connection between stop *a* and *b* at a specific time)
`stop_times.txt`	Yes	Arrival and departure times at a specific stop for a specific trip
`calendar.txt`	Either `calendar.txt` or `calendar_dates.txt`	Information on when trips are repeated (e.g. each Monday and Tuesday during January)
`calendar_dates.txt`	Either `calendar.txt` or `calendar_dates.txt`	Exceptions for the information in `calendar.txt`
`fare_rules.txt`	No	Information on pricing
`shapes.txt`	No	Rules for representing routes on a map
`frequencies.txt`	No	Arrival/departure times defined by time differences between stops
`transfers.txt`	No	Information on transfers between stops
`feed_info.txt`	No	General feed information

The files' contents can be linked using different identifiers. For instance, the `route_id` column of the `routes.txt` file uniquely identifies each route and allows referencing specific routes from other files. For example, trips in the `trips.txt` file can be linked to their corresponding route because each trip row also contains a `route_id`. The relationships between the files as well as the identifiers used are represented in figure 4.2.

The focus of this thesis lies on the representation of information related to a transportation network's topology, i.e. about stations and their connections. These correspond to vertices and edges in the network's graph. This means that mainly routes (i.e. the `routes.txt` file) and stops (i.e. the `stops.txt` file) need to be considered for the Wikibase mapping. However, rows from these two files cannot be linked directly. To be able to

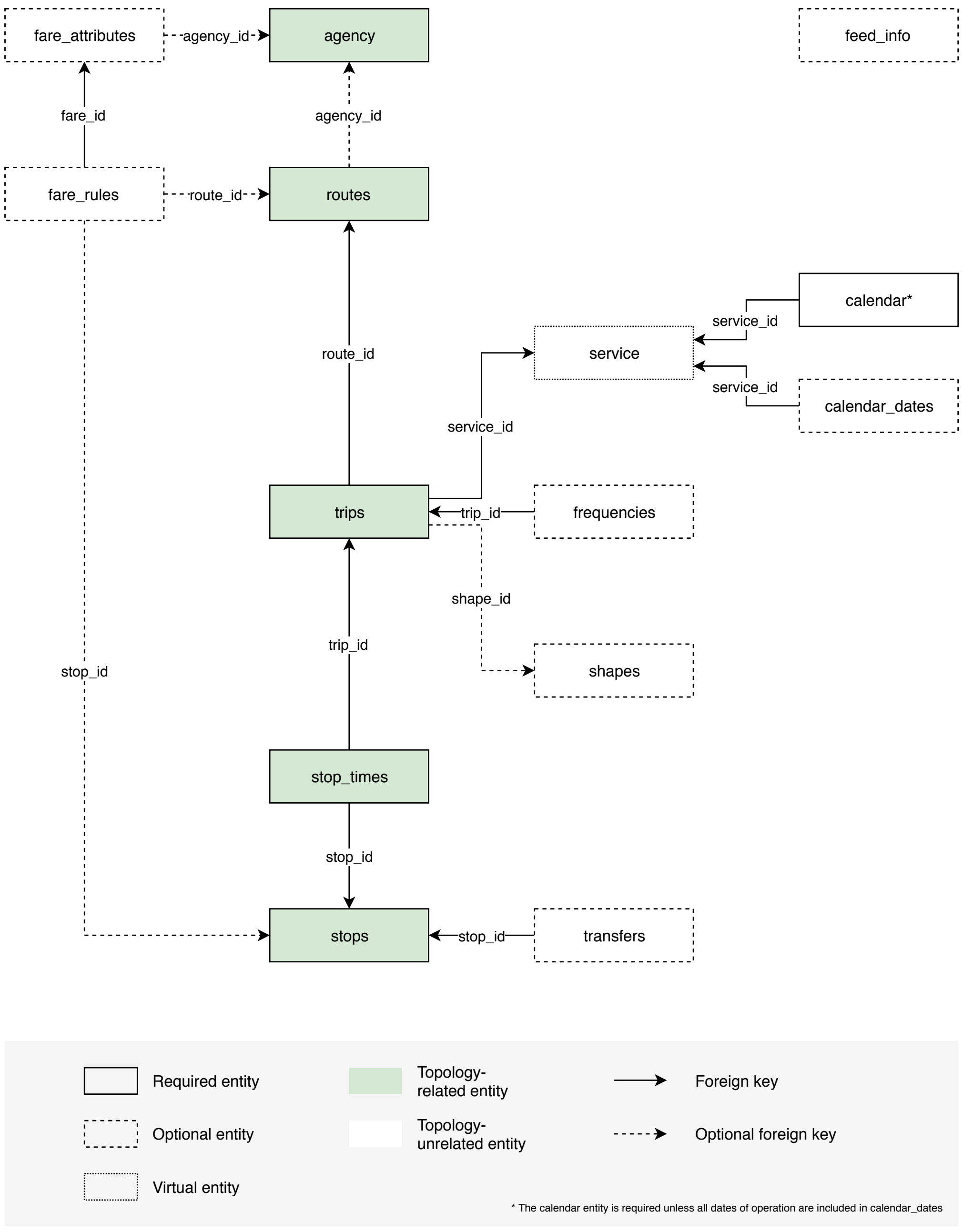


Figure 4.2: GTFS entities and their relations. Entities which are relevant for the transport network's topology are highlighted. Based on 'General Transit Feed Specification Reference' (2018) and Davis (2011)

infer the connections between these two entities, the files `trips.txt` and `stop_times.txt` must be considered as well. Because the `agency.txt` file contains nearly static information and its content provides useful information about routes, it is included in the mapping as well. The remaining files contain data that does not concern the network's topology, but rather timetables, zones, fares, transfers inside stations, and the connections' representation on a map.

To improve the understandability of the data for users and in order to simplify queries, stops' connections are mapped to their parent stations. To take an example, the SBB's data contains stop entries for different platforms inside the Zurich HB station. It is likely that users are not interested in connections for a specific platform but for the entirety of Zurich HB (i.e. the parent station). This is also the reason why the `transfers.txt` file is ignored for the purpose of this mapping, as it is likely to contain mostly information about the transfers between stops which belong to the same parent station.

Figure 4.2 illustrates which files are required and which contain relevant information about the network's topology.

4.2.2 Mapping GTFS Data to the Wikibase Data Model

As mentioned, the entities that are imported from a GTFS feed are stops, routes, and agencies. For each of these entities, a new item is created in Wikibase.

The final mapping is visualised in figure 4.3. It is also available in table form (section A.1.1) and in Turtle notation (listing A.1).

Labels and Descriptions. Item names and descriptions can often be extracted from the item's row in the corresponding GTFS file (e.g. stops have a required `stop_name` and an optional `stop_desc` column).

ID. The row's identifier (`stop_id`/`route_id`/`agency_id`) needs to be imported for the corresponding item to be searchable after its creation (as the combination of the other fields is not necessarily unique over the entire dataset). Unfortunately, Wikibase does not allow the specification of a URI with the creation of an item but assigns these identifiers sequentially. Therefore, the GTFS identifier should be assigned to each item with a claim of the 'external ID' data type.

Membership. To help the user understand what an item represents, it should contain an *instance of* claim which describes what class this item is a member of. For example, an agency item is an instance of the *transport company* Wikidata item.

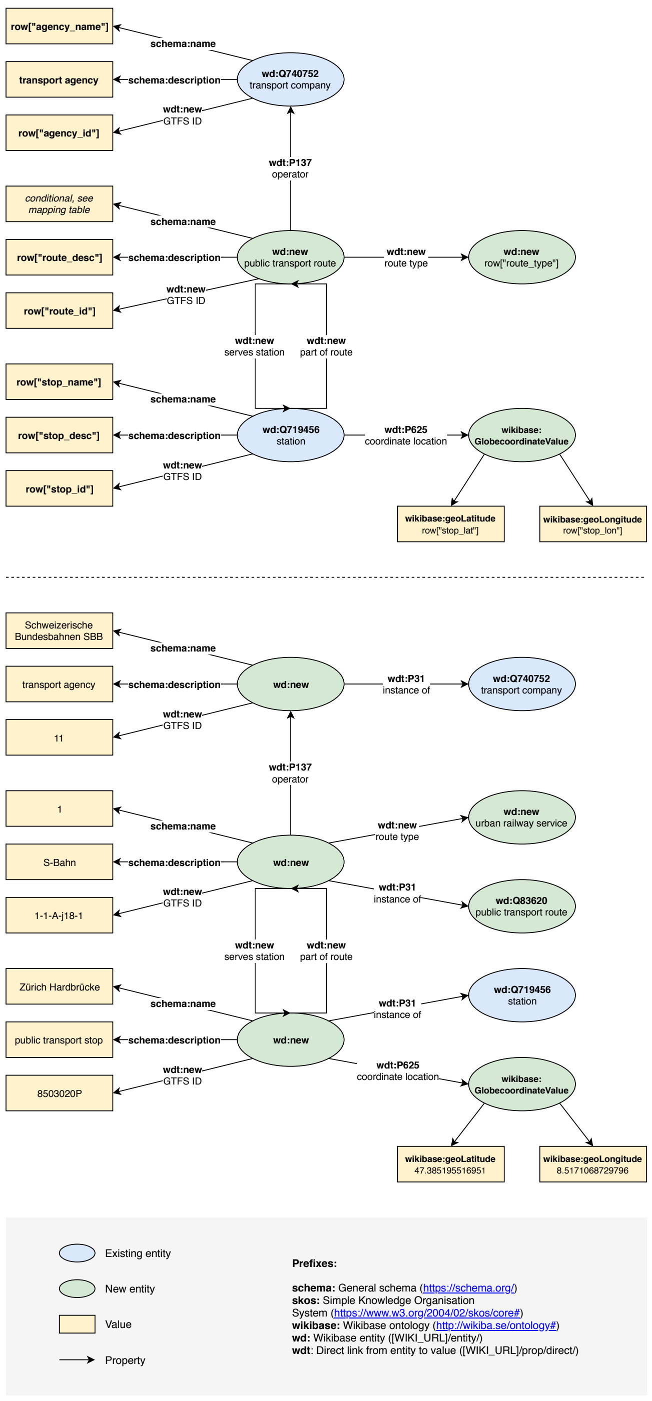


Figure 4.3: GTFS–Wikibase mapping, based on the Wikidata RDF schema ('Wikibase/Indexing/RDF Dump Format', 2019; 'Wikidata:Relation between Properties in RDF and in Wikidata', 2017)

Links Between GTFS Items. Edges, e.g. the relationships between routes and stops, are crucial for the network's topology. In Wikidata, such relationships between items are modelled using statements. For instance, a stop item can be assigned a claim with *part of route* as the property and a route item as the value.

To facilitate the navigation between Wikibase items for users and in order to simplify queries, bidirectional relationships can be used. Since Wikibase's underlying data model is a directed graph and there are no bidirectional properties in Wikibase[2], two edges pointing in opposite directions must be created between the two corresponding vertices.

Because the links between routes and stops are particularly important, bidirectional relationships are used for them in the Wikibase mapping. For the relationships between routes and agencies, unidirectional links might be sufficient as these are not directly related to the transport network's topology and might therefore be queried less often.

Naturally, each route contains multiple *serves station* claims. Because the order in which these stops are served is of interest to the user, it should be represented in the data model. However, the ordering of statements of the same property in Wikibase only depends on their creation dates and cannot be modified directly. Furthermore, their order not meant to carry meaning ('Help:Statements', 2018). Therefore, it makes sense to use qualifiers for representing this information. The `stop_sequence` column from `stop_times.txt` can be used as the value for a qualifier with property *stop index*. The claims may not be listed in order, but it can be deduced from the claims' qualifiers and can therefore also be queried using SPARQL.

Other Statements. Other interesting information from the GTFS feed includes the `route_type`, i.e. the means of public transport that serves the corresponding route, and the coordinate location (`stop_lat`, `stop_lon`) for a public transport stop. This data is also mapped to claims with route type items or instances of the 'globe coordinate' data type as values.

For the route type statements, the Extended GTFS Route Types specification[3] is used, which defines additional route types to the limited set which is part of the GTFS specification.

[2] Discussion on bidirectional Wikidata properties on Phabricator: *https://phabricator.wikimedia.org/T51165*

[3] Extended GTFS Route Types: *https://developers.google.com/transit/gtfs/reference/extended-route-types*

References. While the usage of references is generally highly encouraged, it might be better not to add them to the statements created by the GTFS mapping tool. Every statement would be assigned the same reference value (the source of the GTFS dataset) which could be considered distracting. In case of a Wikibase instance designed to exclusively contain GTFS data, it might make more sense to reference the source in a single place, e.g. in the description on the MediaWiki start page.

Items from Wikidata. Whenever possible, existing Wikidata entities are be used to describe the newly created ones. This ensures compatibility in case the generated items are imported into or linked to Wikidata in the future. If a required entity does not exist on Wikibase, a new one shall be created. In case of an import into Wikidata, the creation of these entities could be suggested to the community.

The used Wikidata entities and the newly created items/properties for the GTFS mapping can be seen in figure 4.3, listing A.1, and the tables in section A.1.1. For the Extended GTFS Route Types, the used/proposed entities are listed in table A.4.

4.3 Import of Static Data

4.3.1 Overview

Python is used for the implementation of the mapping and import scripts because of its popularity and suitability for data processing and analysis. Due to a lack of high-quality libraries[4] for interaction with the MediaWiki/Wikibase API and in order to get more familiar with the Wikibase data model, tools for this task are implemented as well.

The four libraries created for this thesis and their behaviour related to mapping and importing static datasets are described in the following sections. The treatment of dynamic data is explained in section 4.4.

4.3.2 Components

`wikibase-api`. `wikibase-api` is a wrapper library, mostly around the Wikibase extension set for the MediaWiki API[5]. It simplifies the authentication with the API using

[4] Pywikibot (source code: *https://github.com/wikimedia/pywikibot*) is a popular option but, at the time of writing, is more of a collection of scripts than a well-designed library. Furthermore, documentation and examples are scarce and partly outdated.

[5] Wikibase API documentation: *https://www.mediawiki.org/wiki/Wikibase/API* and *https://www.wikidata.org/w/api.php?action=help&recursivesubmodules=1*

OAuth or bot credentials and provides functions for making queries and modifications. The tool handles certain required API parameters internally so the user does not need to worry about them and performs value validations for some parameters. Functions are kept similar to the API requests. In addition, the library is extensively documented and tested due to the critical nature of the tool for the software layers above it. The documentation also includes detailed guides on authentication and the setup of a local Wikibase instance. The aim is to create a straightforward way to get started with the Wikibase API by providing a simpler interface and setup guides.

An example of a Wikibase edit performed using the `wikibase-api` library can be found in listing 4.1.

Listing 4.1: `wikibase-api` usage example: Creating a new Wikibase item with a certain label

```python
from wikibase_api import Wikibase

# Authenticate with Wikibase
wb = Wikibase(config_path="config.json")

# Specify item label in English
content = {
    "labels": {
        "en": {
            "language": "en",
            "value": "item label",
        }
    }
}

# Execute API request
r = wb.entity.add("item", content=content)
```

`python-wikibase`. One layer above, `python-wikibase` wraps around `wikibase-api` with the goal of providing a more user-friendly programming interface. It is no longer kept similar to the API functions but offers an object-oriented abstraction of Wikibase entities and data types as well as functions for querying, creating, modifying, and deleting them. The library provides a hierarchy of classes and custom exception types for improved error handling. Like `wikibase-api`, `python-wikibase` has been tested thoroughly using integration tests.

A UML class diagram of the library can be found in figures 4.4 and 4.5. A usage example of the package can be seen in listing 4.2.

Listing 4.2: `python-wikibase` usage example: Fetching an existing item and adding a claim with a coordinate value

```python
from python_wikibase import PyWikibase

# Authenticate with Wikibase
py_wb = PyWikibase(config_path="config.json")

# Fetch item and "coordinate location" property
item = py_wb.Item().get(entity_id="item label")
prop = py_wb.Property().get(entity_id="coordinate location")

# Create new GeoLocation value (set latitude and longitude)
value = py_wb.GeoLocation().create(1.23, 4.56)

# Create GeoLocation claim
claim = item.claims.add(prop, value)
```

`gtfs-to-wikibase`. `gtfs-to-wikibase` is the package responsible for performing the entity mapping and Wikibase import. The library only requires Wikibase credentials and the path to an unzipped GTFS feed to run. The tool first creates a SQLite database from the feed (GTFS is also structured in a relational way, with the identifiers usable as foreign keys). This database provides a simple means to perform queries. Since a five-way join is needed to connect the required GTFS entities (rows from files `stops.txt`, `stop_times.txt`, `trips.txt`, `routes.txt`, and `agency.txt`) and the script should be able to operate on large datasets like the SBB's, these queries must run efficiently. With the creation of indices for the created tables, SQLite allows for the execution of such queries in a reasonable time. The queries are also used to filter out unwanted data. For instance, stops and their data are mapped to parent stops and only those are kept for the insertion.

After querying all entities and their relations, the relevant data is extracted and the corresponding Wikibase items are created and updated with data. Internally, `gtfs-to-wikibase` uses the `python-wikibase` library for interacting with the Wikibase API.

The reason why no Wikibase items are created for the rows of `stop_times.txt` and

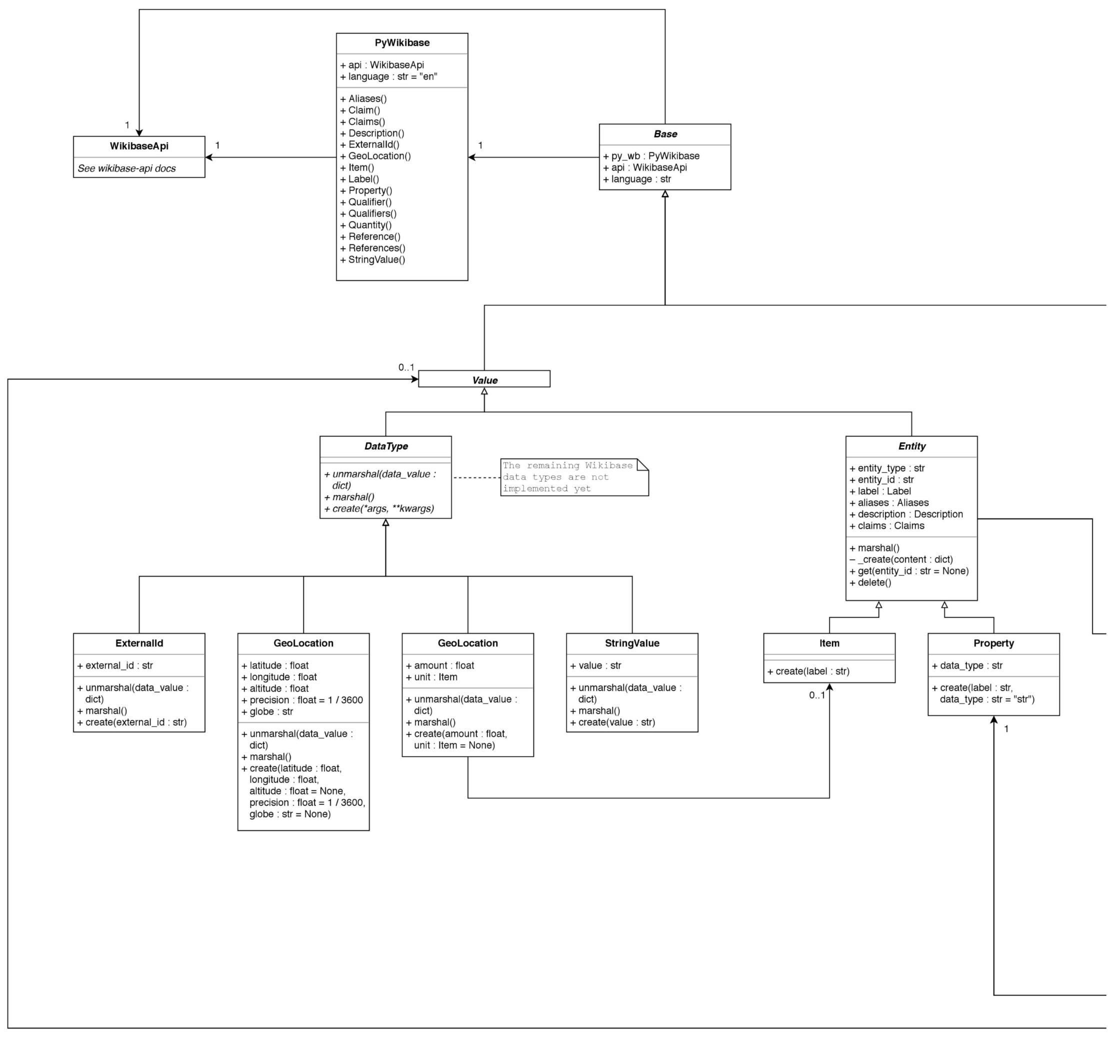


Figure 4.4: UML class diagram of `python-wikibase` (left half)

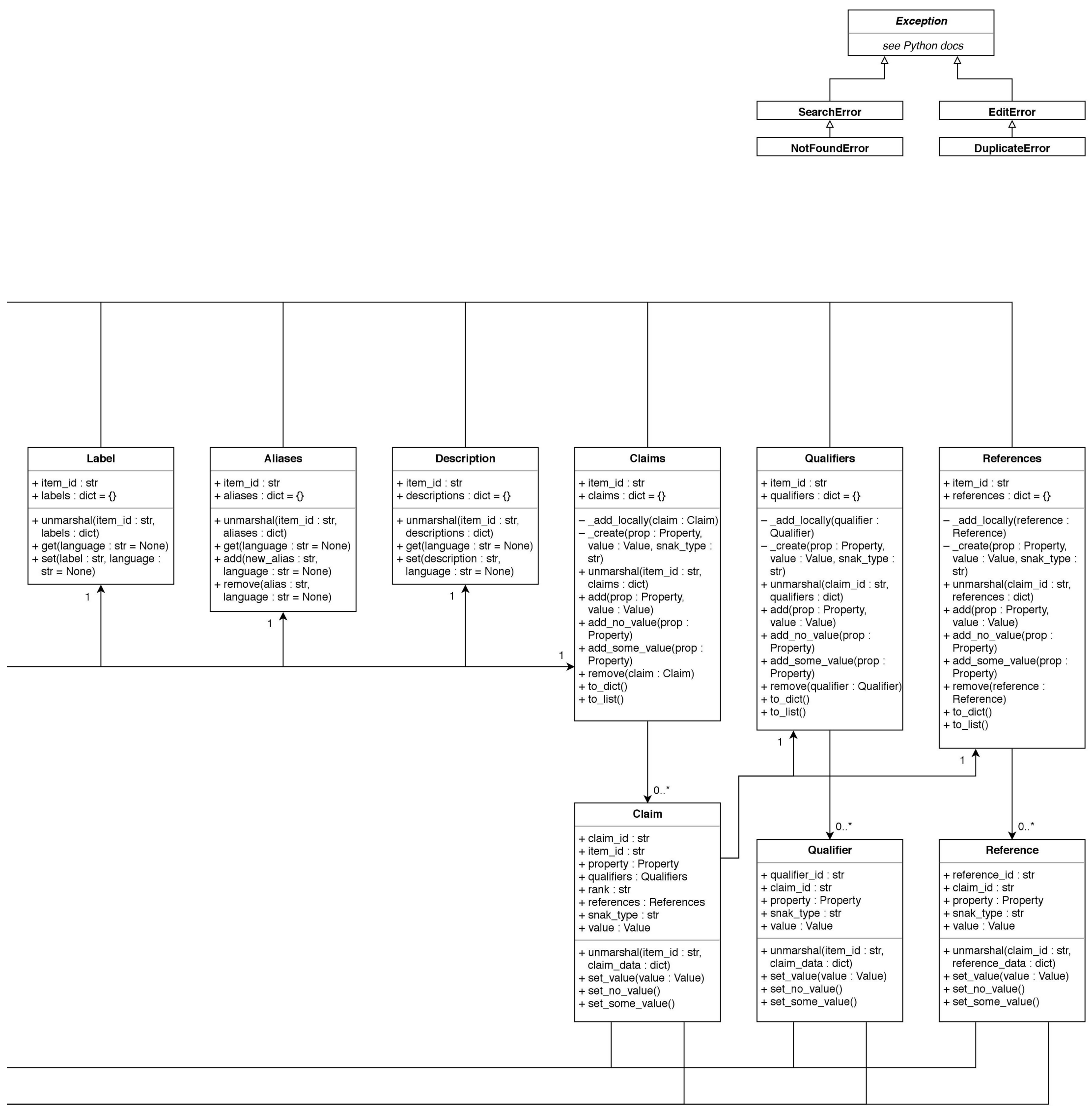

Exception
see Python docs
SearchError
EditError
NotFoundError
DuplicateError
Label
+ item_id : str
+ labels : dict = {}
+ unmarshal(item_id : str, labels : dict)
+ get(language : str = None)
+ set(label : str, language : str = None)
Aliases
+ item_id : str
+ aliases : dict = {}
+ unmarshal(item_id : str, aliases : dict)
+ get(language : str = None)
+ add(new_alias : str, language : str = None)
+ remove(alias : str, language : str = None)
Description
+ item_id : str
+ descriptions : dict = {}
+ unmarshal(item_id : str, descriptions : dict)
+ get(language : str = None)
+ set(description : str, language : str = None)
Claims
+ item_id : str
+ claims : dict = {}
– _add_locally(claim : Claim)
– _create(prop : Property, value : Value, snak_type : str)
+ unmarshal(item_id : str, claims : dict)
+ add(prop : Property, value : Value)
+ add_no_value(prop : Property)
+ add_some_value(prop : Property)
+ remove(claim : Claim)
+ to_dict()
+ to_list()
Qualifiers
+ item_id : str
+ qualifiers : dict = {}
– _add_locally(qualifier : Qualifier)
– _create(prop : Property, value : Value, snak_type : str)
+ unmarshal(claim_id : str, qualifiers : dict)
+ add(prop : Property, value : Value)
+ add_no_value(prop : Property)
+ add_some_value(prop : Property)
+ remove(qualifier : Qualifier)
+ to_dict()
+ to_list()
References
+ item_id : str
+ references : dict = {}
– _add_locally(reference : Reference)
– _create(prop : Property, value : Value, snak_type : str)
+ unmarshal(claim_id : str, references : dict)
+ add(prop : Property, value : Value)
+ add_no_value(prop : Property)
+ add_some_value(prop : Property)
+ remove(reference : Reference)
+ to_dict()
+ to_list()
1
1
1
1
1
1
0..*
0..*
0..*
Claim
+ claim_id : str
+ item_id : str
+ property : Property
+ qualifiers : Qualifiers
+ rank : str
+ references : References
+ snak_type : str
+ value : Value
+ unmarshal(item_id : str, claim_data : dict)
+ set_value(value : Value)
+ set_no_value()
+ set_some_value()
Qualifier
+ qualifier_id : str
+ claim_id : str
+ property : Property
+ snak_type : str
+ value : Value
+ unmarshal(item_id : str, claim_data : dict)
+ set_value(value : Value)
+ set_no_value()
+ set_some_value()
Reference
+ reference_id : str
+ claim_id : str
+ property : Property
+ snak_type : str
+ value : Value
+ unmarshal(claim_id : str, reference_data : dict)
+ set_value(value : Value)
+ set_no_value()
+ set_some_value()

Figure 4.5: UML class diagram of `python-wikibase` (right half)

`trips.txt` is that there is a substantial number of them and that they only contain information that – presented alone – is not of interest to the end user. Furthermore, their import would result in too many items being created in Wikibase so the script could no longer run in acceptable time. This is because the Wikibase API does not allow multiple items to be created with a single API call, making the creation of a large number of items exceedingly slow.

`gtfs-to-wikibase` can be used as a command-line interface or as a Python library.

`sbb-wikibase`. The fourth and most application-specific library is `sbb-wikibase`. It is a script which downloads the SBB's most recent GTFS feed, extracts it to a directory, and runs `gtfs-to-wikibase` on the files. `sbb-wikibase` is designed to be used as a command-line interface.

4.3.3 Deployment

The goal of a transport-specific Wikibase instance is achieved using `wikibase-docker`[6]. It provides Docker images for setting up a custom Wikibase instance. `sbb-wikibase` also provides a Docker image of the implemented software designed for usage with `wikibase-docker`. However, it could also be configured to perform the required insertion on a live instance like Wikidata.

4.4 Import of Dynamic Data

4.4.1 Updating Data in Wikibase

In a second step after the mapping, the software needs to be able to handle changes in the GTFS dataset and update the Wikibase entities accordingly. This functionality is provided by the `gtfs-to-wikibase` package.

When the script is provided with a new GTFS dataset after having performed an initial insertion, it compares the current and the previously processed datasets. If they are different, the software queries the changes, i.e. the additions, removals, and updates to items, claims, and qualifiers. Having obtained this result, the tool looks up the changed entities in Wikibase and creates, deletes, or updates them.

[6] `wikibase-docker` source code: *https://github.com/wmde/wikibase-docker*

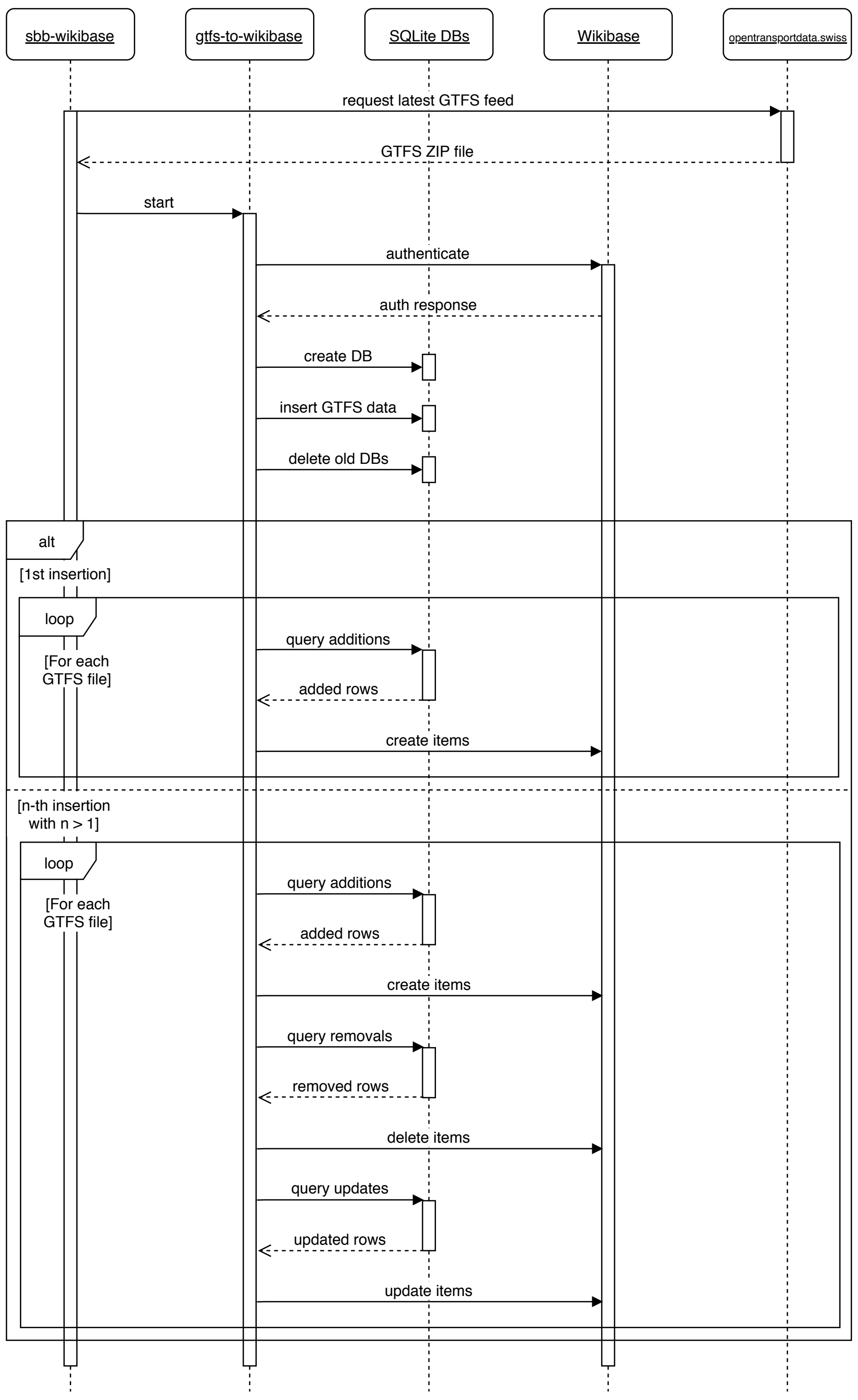


Figure 4.6: UML sequence diagram of the GTFS updater software. The processes are described in a simplified and abstract way

4.4.2 Punctuality Information

Calculation. Besides their GTFS datasets, the SBB also make data on all effective public transport trips of the previous day publicly available, which they call 'actual data' datasets[7]. This data includes expected as well as actual arrival and departure times for the connections. This allows for an analysis of the connections' delays and the addition of its results into the Wikibase items created from the GTFS feeds. It is important to note, however, that the exact arrival and departure times are only available for a small subset of the trips in the dataset. Many rows only contain predictions. These are filtered out for the purpose of this analysis. Furthermore, while the `stop_id`s from the GTFS datasets can be matched with the station identifiers from the punctuality dataset, the same is not possible for the trip identifiers. This means that even though the data is available for stops and trips, punctuality information can only be added to stops. This data informs users about how much connections arrived and departed early or late at the stops on the previous day.

It makes sense to run the updating script once a day since the punctuality datasets are published in the same interval.

Integration into Wikibase. Various options exist for representing punctuality information in the MediaWiki UI for stop items. Because it is a goal of this thesis to make the imported data as accessible and understandable for users as possible, a small survey for identifying user preferences is performed.

The aim of conducting this survey is to answer the following questions:

1. How do potential users define punctuality, i.e. how much does a connection need to be delayed to be considered late?
2. Should this definition of punctuality be used to display the relative number of delayed connections for a stop or do users prefer more exact information, e.g. the average delay at the station?
3. Are users interested in the arrival delay, the departure delay, or both?

The full survey questions can be found in section A.5.1. The results are analysed in section 6.

[7] SBB 'actual data' open datasets: *https://opentransportdata.swiss/en/dataset/istdaten*

5 Representation of Dynamics

5.1 Update History and Summarisation

One of the goals of this thesis is to provide users and machines with a way to get an impression of the dynamics of the imported data, i.e. of what has changed over time and how much. Such representations of dynamics can be created on two levels: First, a report can be created for the level of a single Wikibase item, where it is reasonable to have a history of changes performed on the item. Secondly, the entire dataset can be represented, where updates could be aggregated or summarised in a way that makes the nature of the dynamics understandable and makes potential trends visible for users and machines.

5.2 Dynamics of a Single Entity

On the page of a single item or property (see example in figure A.1), the MediaWiki interface shows users a 'View History' tab where the change history for this particular entity is displayed (see example in figure A.2). Using this site, users can inform themselves about the size and nature of changes, and MediaWiki even allows them to compare the entity's contents between two points in time. For instance, in the case of the SBB dataset, the history view of a public transport stop item lets users inspect when a certain route stopped serving this particular stop (i.e. when the corresponding *part of route* claim was removed).

5.3 Dynamics of a Dataset

What MediaWiki and Wikibase do not provide, however, is an *overview* of the changes that were performed on the dataset. Information of this kind – e.g. the total number of created, deleted, and updated entities or claims – might be of general interest, also for other linked open datasets. For this reason, a way to represent such information in a machine-readable way is developed for this thesis.

Datasets and their contents are often described using dedicated RDF ontologies. Popular options include DCAT and VoID. DCAT is used to describe data catalogues with the contained datasets and distributions (Maali, DERI, NUI Galway, Erickson & Tetherless World Constellation (RPI), 2014), while VoID is suitable for specifying datasets (especially linked ones) and their metadata and links (Alexander, Cyganiak, Hausenblas & Zhao, 2011). For an application of the two vocabularies to the SBB dataset, see listings A.2 and A.3.

While these ontologies are informative about the nature of datasets, there currently does not seem to be a way to model information about changes to a dataset using these vocabularies. This functionality would allow summarising the dynamics of a dataset as desired. For this purpose, a new vocabulary is created as part of this thesis which is designed to be used in conjunction with DCAT. This vocabulary, hereinafter called 'dynamics vocabulary' and abbreviated with `dyn`, defines the following entities:

- **Classes:**
  - `dyn:Entity`: Item class which shall be monitored for updates. In the case of the SBB dataset, the instances 'Agency', 'Route', and 'Stop' are created
  - `dyn:VersionHistory`: History of updates to a dataset, could be added to an instance of the `dcat:Dataset` class
  - `dyn:Version`: Specific dataset version in the dataset's version history. Contains links to entity versions and update statistics for the entire dataset
  - `dyn:EntityVersion`: Class which contains update statistics for a `dyn:Entity`. Assigned to a `dyn:Version` of the dataset
- **Properties:**
  - `dyn:versions`: Assigns a `dyn:VersionHistory` to an instance of the `dcat:Dataset` class
  - `dyn:version`: Used to list the different versions in a version history
  - `dyn:entityVersion`: Links a `dyn:EntityVersion` to a `dyn:Version`

- `dyn:additionsAbs`, `dyn:deletionsAbs`, `dyn:updatesAbs`: Number of additions/deletions/updates made in the corresponding dataset update, either for the entire dataset or for a specific `dyn:Entity`
- `dyn:additionsRel`, `dyn:deletionsRel`, `dyn:updatesRel`: Percentage of updated entities during the corresponding dataset update, either compared to all entities in the dataset or to the entities of a specific `dyn:Entity` class

To get an impression of the vocabulary being applied to the SBB dataset, see figures 5.1, A.3, and A.4, as well as listing A.4.

By parsing the information provided using the dynamics vocabulary, software can interpret when a dataset was changed, how many changes were performed, and which entities were affected to what extent. Using this information, especially the relative change counters, software can also infer which entity types are likely to undergo changes in the future. The knowledge of such properties could be useful for LOD use cases like data caching and indexing (Dividino, Gottron, Scherp & Gröner, 2014).

To be able to generate the statistics required by the dynamics vocabulary, `sbb-wikibase` includes a module for automatically keeping track of the number and nature of changes being made. After a database update completes, this information is saved to disk. To make the understanding of this data easier for humans, the `sbb-wikibase` script generates or updates the triples which are part of the vocabulary as Wikibase entities or claims, allowing users to look up the information in the familiar MediaWiki interface.

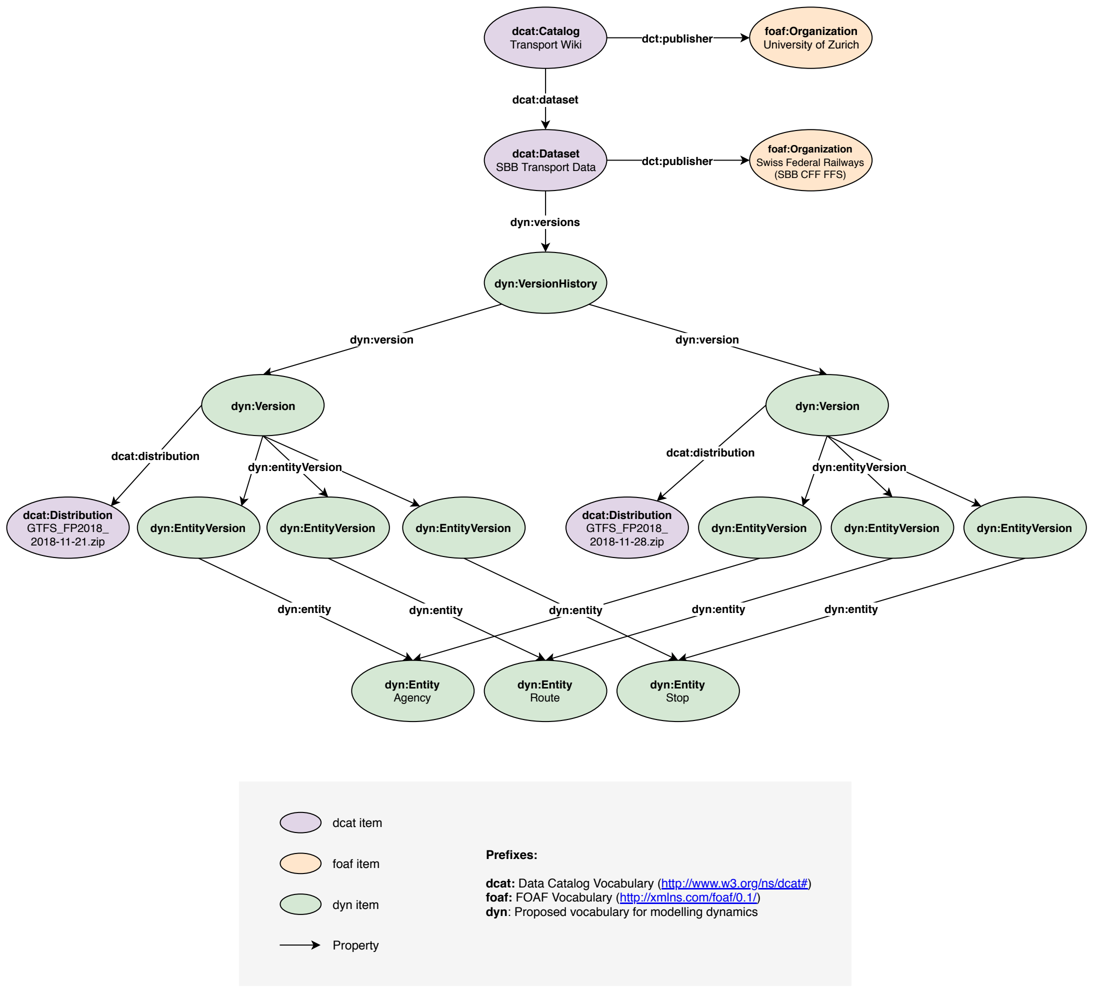


Figure 5.1: Extension to the DCAT vocabulary for modelling the dynamics in a LOD dataset, applied to the SBB data. Only items and the corresponding properties are modelled; literals are ignored

6 Analysis

6.1 Overview

The software developed in this thesis (described in section 4) was executed on the SBB's GTFS and punctuality datasets for most of 2018, specifically on the datasets for every day between 03/01/2018 and 01/11/2018[1]. A Jupyter Notebook[2] with various descriptive statistics and plots was then created to analyse the outcome of these insertions. The notebook can be found in section A.7. The goal of the analysis is to understand how the SBB's network evolves over time and whether the degree to which it changes varies over the course of a year.

Similar statistics were also created for the punctuality datasets. Additionally, the results of the punctuality survey, which help with the decision on how to best represent information on connection delays for users in Wikibase, are shown in what follows.

The results of the analysis are presented in this chapter and discussed in chapter 7.

6.2 Analysis of Network Dynamics

The first part of the analysis deals with the dynamics of the SBB transportation network. Various plots show the changes for items (stops, routes, and agencies) as well as stop–route links (the *part of route* and *serves station* claims that connect these two types of items) over time. Furthermore, several plots show how much the item types contribute

[1] The SBB have not made all datasets about 2018 publicly available. The GTFS dataset from 28/03/2018 and the 'actual data' dataset from 24/05/2018 are missing. Furthermore, the GTFS dataset from 15/08/2018 is corrupted.

[2] Project Jupyter: *https://jupyter.org*

to the total number of added/removed/updated items over time. In the figures, the dates of major timetable changes are highlighted. For the analysed time period, these are the start and the end date of the summer timetable.

It is noticeable from the plots that among the three Wikibase item types – stops, routes, and agencies – routes were changed most frequently by far (see the 'Changes over Item Types' section in the notebook). Such changes to route items automatically propagated as changes to the stop–route link claims.

The total number of changes to the network rose slightly during the year (see 'Total number of changes'). One point in the graphs that attracts attention is a significant spike in the item additions on 11/09/2018. Most of these added items were stops.

What is noteworthy from the graph on item updates is that, apparently, only routes were updated in the SBB dataset (see 'Distribution of updates over entity types'). Information about agencies and stops remained constant during the year, even though such entities were frequently added and removed.

6.3 Analysis of Punctuality

6.3.1 Survey Results

What follows are the results of three survey questions conducted on punctuality (introduced in section 4.4.2). The survey was conducted on a small sample ($n = 18$) as it was simply intended to help with the understanding of how users would like punctuality information to be represented in Wikibase.

1. With outliers removed (using the interquartile range), users on average define connections as late if they arrive 3.5 minutes (median 3 minutes) after the scheduled time.

2. A majority (61.1%) of participants is only interested in being presented the average delay per connection. Some people additionally want to know the percentage of delayed connections (27.8%), but nobody is interested in the percentage alone.

3. Almost all participants (88.9%) associate delay with departure time. For 44.4% of all participants, the arrival time is of equal importance. 11.1% of people only consider the arrival time.

Based on these results, the average delay of connections for both arrival and departure is added to the stop items in Wikibase.

6.3.2 Punctuality over Time

The second part of the data analysis performed using the Jupyter Notebook shows statistics on the inserted punctuality information. First, descriptive statistics on arrival and departure delays over the analysed period are calculated. Secondly, the difference between arrival and departure delays on the previous day is analysed and a list of stops where the delay increased is shown.

The notebook shows that over the insertion period, connections on average arrived 83 seconds late at the station and left 140 seconds late (see ‘Average daily delay per station’). The average arrival delay rose slightly over the course of the year whereas the departure delay remained relatively constant (see ‘Average daily delay over time’).

7 Discussion

7.1 Results

7.1.1 Network Dynamics

What follows is a discussion of the data analysis results shown in chapter 6.

In some plots, the graphs drop to zero at certain points in time. The reason for the gaps is that datasets about these dates are not or no longer available on the SBB's open data platform.

It is somewhat surprising that the number of changes is not significantly different during the summer timetable of 2018, as more construction work and temporary timetable changes occurred during that time.

The SBB do not guarantee that the identifiers of routes and trips remain the same between dataset versions[1]. However, these IDs are required for the identification of changes between two versions. If a route identifier changes from one version to the next, the software will count one deletion and one addition. The results of the analysis related to route dynamics, which showed a very high number of changes, therefore need to be interpreted cautiously, as these ID changes are responsible for the majority of what the software considers as dynamics. The same applies to the changes in stop–route links, as changes to routes will also result in changes to these claims.

The rising number of changes in the network over the year 2018 could be related to structural changes or an expansion of the transport network. It would be interesting to relate this information to a yearly report by the SBB. Unfortunately, such data does not seem to be publicly available yet for 2018.

[1] See FAQ on Open Data Platform Swiss Public Transport: *https://opentransportdata.swiss/de/faq*

In the data analysis, it was noted that apparently, the SBB do not update agency and stop items, but only routes. It makes sense that agency information does not receive updates since the only field whose change would manifest in Wikibase is the agency's name, which is very unlikely to be changed. For stops, however, updates would make sense. For instance, when a stop is moved temporarily due to construction work, its coordinates would need to be updated. It is possible, however, that the SBB do not record such temporary changes in their databases.

7.1.2 Punctuality

Because the structure and contents of the SBB's 'actual data' datasets changed during 2018, punctuality data is missing for the majority of the year's first couple of months.

The survey on punctuality showed that people on average consider a connection to be late if it is 3.5 minutes delayed. According to the analysis of the SBB's datasets, the average connection is within that time (arrivals are 83 seconds late and departures are 140 seconds late) and can therefore considered punctual. In their own datasets, the SBB define trains as delayed if they are more than 3 minutes late, which is very similar to the time specified by the survey participants.

It is notable that the average arrival delay increased during 2018, whereas the average departure delay did not change significantly. It would be interesting to relate this pattern to the SBB's knowledge of their network.

The departure delay is generally higher than the arrival delay, which might be explainable by the connections making up time between the stations.

Delays did not increase significantly while the summer timetable 2018 was in place, which might indicate that the temporary network changes were well planned and executed by the SBB.

7.1.3 Dynamics Vocabulary

Using the developed vocabulary for modelling dataset dynamics, users and software can now obtain summary information about the changes that were made to the dataset at different points in time. While this approach is suitable for machines because they can easily parse the generated RDF entities and extract the information of interest, the solution is not ideal for humans. When created as Wikibase entities like in the transport Wikibase instance, the vocabulary entities are difficult to discover and it is not intuitive that the change information is distributed over many nodes. Ideally, the MediaWiki interface would contain a dedicated site for showing information on the dataset dynamics which contains all relevant data in one place. This could be represented e.g. in table form when visited by a human and in the described RDF structure when visited by a machine. The type of user could be identified using the `Accept` HTTP request header, as it is already done for the entity pages.

7.2 Challenges

One major challenge of this thesis was the planning and time estimation related to the software implementation. While designed as a thesis with a focus on programming, the software development part required more time than anticipated, leaving less time for other aspects like the analysis of the graph data. Reasons are difficulties with estimating the scope of the thesis' tasks because of unfamiliarity with some of the topics at the beginning of the work and the need to develop some of the lower-level software libraries due to a lack of high-quality tools for the purpose.

What also proved to be a challenge is the work with datasets of such a large size. The SBB's datasets contain millions of rows, naturally making computations more lengthy. While the majority of the development could be performed on test datasets, the software still needed to be tested regularly on the SBB's data, which is a time-consuming process. Furthermore, discrepancies of the SBB's datasets from the official GTFS specification frequently took some time to be identified and made parts of the development more complicated.

In addition, the design of the MediaWiki API made the development of the lower-level libraries somewhat difficult. At the time of this writing, not all API requests and data types are documented equally well, the API is unforgiving regarding malformed requests, and the provided error messages are often unspecific and sometimes misleading. For this reason, the abstraction of such requests using the `wikibase-api` and `python-wikibase` libraries was useful and time-saving in the later stages of development. The packages might therefore be of interest to other developers looking to work with the Wikibase API.

7.3 Limitations

Some constraints apply to the resulting Wikibase representation of the SBB's datasets due to some of their attributes:

- As mentioned in section 4.4.2, trips from an 'actual data' file cannot be matched with trips in a GTFS feed because different identifiers are used. Currently, there does not seem to be a way to match these identifiers using publicly available information.

- Some routes have corresponding trips which pass different stops. Such trip variations are also referred to as branches[2]. For instance, a bus may skip a certain stop at a certain time of day or alternate between two terminal stops for one trip direction. Such route branches cannot be mapped to Wikibase because they do not fit in the data model. In order to obtain valid stop sequences and indices in Wikibase, the main branch is identified in `gtfs-to-wikibase`, i.e. the sequence of stops which occurs most often for the corresponding route. Only this main branch is then imported and its stop indices are used. To avoid a significant performance loss because of the branch identification, it is integrated into the SQL route query. The query uses SQLite's `group_concat` function to construct unique branch identifiers from the stop IDs and return the information for the branch whose branch identifier appears most often.

- To make the identification of the main route branch reliable[3], only trips in the default travel direction (`direction_id != 1`) are considered. However, the SBB's dataset contains routes for which all trips have `direction_id == 1`, even for trips of bidirectional routes. Because of this implementation, some information may be missing in the mapping of the SBB's data to Wikibase. It is possible that a change in this direction information caused the spike of stops in September 2018 that was pointed out in the data analysis.

A general limitation of the current implementation is that the addition of previously deleted entities cannot be detected. This is because only the current and previous datasets are compared. If an entity is added again after being deleted in an older dataset version, a new Wikibase item is created and therefore, the version history of the old item is not restored.

[2] Branches are described in on the GTFS Best Practices website: *https://gtfs.org/best-practices/#branches*

[3] When the main route branch has the same number of trips in both directions, the selection of one of them for the import into Wikibase is indeterminate. If at time t_1, the route branch with `direction_id == 0` were identified as the main branch, and at time t_2, the same branch with `direction_id == 1` were used, this would incorrectly be counted as a change in the dataset. To avoid this behaviour, only trips with `direction_id != 1` are considered.

7.4 Future Directions

The SBB's intention is to integrate a subset of their open data into the Wikidata ecosystem. In this thesis, the foundation of this process was built by creating a way to map and import GTFS data into Wikibase. To expand upon the work done in this thesis, the created transport Wikibase instance could be linked to Wikidata. For instance, the entities' IDs in the transport instance could be added as 'external ID' claims to the corresponding Wikidata entries. Alternatively, the import script could be applied to Wikidata directly. Integrating this information into Wikidata would increase the discoverability of the SBB data and guarantee that up-to-date information about the SBB's transportation network can be found from Wikidata.

The analysis of dynamics performed in this thesis is mainly concerned with the absolute and relative number of changes to the transport dataset over time. This approach allows for a quantitative comparison of the data and the contained entity types between two points in time – in this analysis, between a specific day and the previous day (for punctuality data) or week (for network data). However, such an approach does not truly measure the dynamics between more than two points in time (Dividino et al., 2014). Instead, one could expand upon such snapshot analyses by building a continuous function modelling these dynamics and possibly applying a decay function. This would result in richer information on dynamics. In the next step, such an approach could be applied to a public transport dataset like the SBB's.

The punctuality analysis could also be expanded upon, e.g. by studying demographic differences in the perception of punctuality. Additionally, with more significant results related to dynamics as mentioned above, their relationship with punctuality could also be analysed further.

Lastly, the data analysis in this thesis was initially planned to analyse the relation between network dynamics and punctuality. Due to time constraints, this was not possible, and the aforementioned limitations of the SBB's datasets would make significant results unlikely. In the future, the developed software could be applied to other suitable datasets and the dynamics and punctuality of that transportation network could be analysed for a possible correlation.

8 Conclusions

In this thesis, a set of software tools was developed with the purpose of extracting topology information from GTFS feeds and mapping and importing this data into an instance of Wikibase. Due to frequent changes in such datasets, the software was built to be able to detect changes between two versions and apply the updates to the Wikibase entities. In order to make the dynamics of such datasets more understandable, an extension to the DCAT vocabulary was created which allows modelling summarised change information. Using this ontology, changes made during a dataset update can be made accessible for humans and machines.

The described software was executed on the datasets of the Swiss Federal Railways (SBB), the entities related to the vocabulary extension were created in Wikibase, and the generated update summaries were used as the basis of a simple descriptive data analysis. Due to certain attributes of the datasets and time constraints, the results of this analysis are limited.

In future work, the developed tools could be applied to a longer period of time or other transportation datasets, the mapped SBB datasets could be integrated into Wikidata, and the data analysis could be extended to test for a correlation between network dynamics and punctuality.

A

Appendix

A.1 GTFS–Wikibase Mapping

A.1.1 Mapping tables

`agency.txt`

Table A.1: Mapping of `agency.txt` GTFS row to Wikibase item

Type	**Condition**	**Property**	**Value**
Label			`row["agency_name"]`
Description			`"transport agency"`
Statement		GTFS ID (new)	`row["agency_id"]`
Statement		instance of (P31)	transport company (Q740752)

`routes.txt`

Table A.2: Mapping of `routes.txt` GTFS row to Wikibase item

Type	**Condition**	**Property**	**Value**
Label	`If row["route_short_name"] and row["route_long_name"]`		`row["route_long_name"] + " (" + row["route_short_name"] + ")"`
Label	`Elif row["route_long_name"]`		`row["route_long_name"]`
Label	`Elif row["route_short_name"]`		`row["route_short_name"]`
Description	`If row["route_desc"]`		`row["route_desc"]`

Description	`Else`		`"public transport route"`
Statement		GTFS ID (new)	`row["route_id"]`
Statement		instance of (P31)	thoroughfare (Q83620)
Statement		route type (new)	entity corresponding to a GTFS route type
Statement	`If row["agency_id"]`	operator (P137)	agency
Statement		serves station (new)	stop (usually multiple statements, see stop_times.txt
Qualifier		stop index (new)	`row["stop_sequence"]` (for each stop statement described in the row above)

As route types, one of the eight general route types from the GTFS specification[1] or one of the more specific route types defined in the Extended GTFS Route Types extension[2] can be used. The SBB use the latter in their datasets.

`trips.txt`

No Wikibase entity needs to be generated (not relevant for network topology). However, the mapping of the `row["trip_id"]` and the corresponding `row["route_id"]` must be considered so stops can be assigned to routes.

`stops.txt`

Stops with `row["location_type"] == 1`, i.e. station entrances/exits, and stops where `row ["parent_station"]` is defined must be ignored.

Table A.3: Mapping of `stops.txt` GTFS row to Wikibase item

Type	**Condition**	**Property**	**Value**
Label			`row["stop_name"]`
Description	`If row["stop_desc"]`		`row["stop_desc"]`
Description	`Else`		`"public transport stop"`
Statement		GTFS ID	`row["stop_id"]`
Statement		instance of (P31)	station (Q719456)

[1] GTFS route types: *https://developers.google.com/transit/gtfs/reference/#routestxt*
[2] Extended GTFS Route Types: *https://developers.google.com/transit/gtfs/reference/extended-route-types*

Statement		coordinate location (P625)	`row["stop_lat"], row ["stop_lon"]`
Statement		part of route (new)	route

`stop_times.txt`

No Wikibase entity needs to be generated (not relevant for network topology). Using this file, stops can be associated with trips and therefore with routes.

Other GTFS files

No Wikibase items need to be generated for rows of the following files because they are not relevant for the transport network's topology:

- `calendar.txt`
- `calendar_dates.txt`
- `fare_attributes.txt`
- `fare_rules.txt`
- `shapes.txt`
- `frequencies.txt`
- `transfers.txt`
- `feed_info.txt`

A.1.2 Turtle Representation

Listing A.1: GTFS–Wikibase mapping in Turtle

```
@prefix rdfs: <https://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <https://schema.org/> .
@prefix skos: <https://www.w3.org/2004/02/skos/core#> .
@prefix wikibase: <http://wikiba.se/ontology#> .
@prefix wd: <[WIKI_URL]/entity/> .
@prefix wdt: <[WIKI_URL]/prop/direct/> .
@prefix wdv: <[WIKI_URL]/value/> .


# ITEMS
# -----

# Agency
wd:Q[ID_AGENCY] a wikibase:Item ;
  rdfs:label "Schweizerische Bundesbahnen SBB"@de ;
  schema:description "transport agency"@en ;
  wdt:P[ID_GTFS_ID] "11"@de ;
  wdt:P31 wd:Q740752 .

# Route
wd:Q[ID_ROUTE] a wikibase:Item ;
  rdfs:label "1"@de ;
  schema:description "S-Bahn"@de ;
  wdt:P[ID_GTFS_ID] "1-1-A-j18-1"@de ;
  wdt:P31 wd:Q[ID_PUBLIC_TRANSPORT_ROUTE] ;
  wdt:P[ID_ROUTE_TYPE_PROP] wd:Q[ID_ROUTE_TYPE] ;
  wdt:P137 wd:Q[ID_AGENCY] ;
  wdt:P[ID_SERVES_STATION] wd:Q[ID_STOP] .
  # The above statement's qualifier (route index) is omitted

# Stop
wd:Q[ID_STOP] a wikibase:Item ;
  rdfs:label "Zürich Hardbrücke"@de ;
  schema:description "public transport stop"@en ;
  wdt:P[ID_GTFS_ID] "8503020P"@de ;
  wdt:P31 wd:Q719456 ;
  wdt:P625 wdv:[ID_GEO] ;
  wdt:P[ID_PART_OF_ROUTE] wd:Q[ID_ROUTE] .

# "public transport route" item
wd:Q[ID_PUBLIC_TRANSPORT_ROUTE] a wikibase:Item ;
  rdfs:label "public transport route"@en .

# GTFS extended route type
wd:Q[ID_ROUTE_TYPE] a wikibase:Item ;
  rdfs:label "urban railway service"@en .


# PROPERTIES
# ----------

# "GTFS ID" property
wd:P[ID_GTFS_ID] a wikibase:Property ;
```

```
  rdfs:label "GTFS ID"@en ;
  wikibase:propertyType wikibase:ExternalId .

# "route type" property
wd:P[ID_ROUTE_TYPE_PROP] a wikibase:Property ;
  rdfs:label "route type"@en ;
  wikibase:propertyType wikibase:WikibaseItem .

# "serves station" property
wd:P[ID_SERVES_STATION] a wikibase:Property ;
  rdfs:label "serves station"@en ;
  wikibase:propertyType wikibase:WikibaseItem .

# "stop index" property (used as a qualifier in wd:Q[ID_ROUTE]) is
    omitted

# "part of route" property
wd:P[ID_PART_OF_ROUTE] a wikibase:Property ;
  rdfs:label "part of route"@en ;
  wikibase:propertyType wikibase:WikibaseItem .


# VALUES
# ------

# Globe coordinate of stop
wdv:[ID_GEO] a wikibase:GlobecoordinateValue ;
  wikibase:geoLatitude "47.385195516951"^^xsd:double ;
  wikibase:geoLongitude "8.5171068729796"^^xsd:double ;
  wikibase:geoPrecision "0.000277778"^^xsd:double ;
  wikibase:geoGlobe <https://www.wikidata.org/entity/Q2> .
```

A.1.3 Representation in DCAT and VoID Vocabularies

Listing A.2: GTFS–Wikibase mapping in DCAT

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
```

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .


:uzh a foaf:Organization ;
  rdfs:label "University of Zurich" ;
  foaf:homepage <https://www.uzh.ch> .

:sbb a foaf:Organization ;
  rdfs:label "Swiss Federal Railways (SBB CFF FFS)" ;
  foaf:homepage <https://www.sbb.ch> .

:catalog a dcat:Catalog ;
  dct:title "Transport Wiki" ;
  rdfs:label "Transport Wiki" ;
  foaf:homepage <[WIKI_URL]> ;
  dct:subject <http://dbpedia.org/resource/Public_transport> ;
  dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
  dct:publisher :uzh ;
  dcat:dataset :dataset-sbb .

:dataset-sbb a dcat:Dataset ;
  dct:title "SBB Transport Data" ;
  dcat:keyword "public-transport", "transport", "network", "gtfs", "sbb" ;
  dct:issued "2018-11-21"^^xsd:date ;
  dct:modified "2018-11-21"^^xsd:date ;
  dcat:contactPoint <mailto:data@sbb.ch> ;
  dct:publisher :sbb ;
  dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
  dcat:landingPage <https://opentransportdata.swiss/dataset/timetable
      -2018-gtfs> ;
  dcat:distribution :gtfsfp20182018-11-21.zip .

:gtfsfp20182018-11-21.zip a dcat:Distribution ;
  dcat:downloadURL <https://opentransportdata.swiss/dataset/b408f747
      -9838-4c05-bb98-10dac3996f17/resource/50066861-ac4e-41ff-9015-5
      fefda985952/download/gtfsfp20182018-11-21.zip> ;
  dct:title "GTFS_FP2018_2018-11-21.zip" ;
  dcat:mediaType "zip" ;
  dcat:byteSize "86303506"^^xsd:decimal .
```

Listing A.3: GTFS–Wikibase mapping in VoID

```
@prefix void: <http://rdfs.org/ns/void#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .


:DBpedia a void:Dataset ;
  dct:title "Transport Wiki" ;
  rdfs:label "Transport Wiki" ;
  foaf:homepage <[WIKI_URL]> ;
  dct:subject <http://dbpedia.org/resource/Public_transport> ;
  dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
  dct:publisher :uzh ;
  void:sparqlEndpoint <[SPARQL_URL]> ;
  void:uriLookupEndpoint <[WIKI_URL]/index.php?search=&search=qt> ;
  void:uriSpace "[WIKI_URL]/entity/" ;
  void:vocabulary <http://www.w3.org/ns/dcat#> ;
  void:vocabulary <https://schema.org/> ;
  void:vocabulary <https://wikiba.se/ontology#> ;
  void:vocabulary <[WIKI_URL]/entity/> ;
  void:vocabulary <[WIKI_URL]/prop/direct/> ;
  void:vocabulary <[WIKI_URL]/value/> .

:uzh a foaf:Organization ;
  rdfs:label "University of Zurich" ;
  foaf:homepage <https://www.uzh.ch> .

:sbb a foaf:Organization ;
  rdfs:label "Swiss Federal Railways (SBB CFF FFS)" ;
  foaf:homepage <https://www.sbb.ch> .
```

A.2 Extended GTFS Route Types

Table A.4: Mapping of Extended GTFS Route Types ('Extended GTFS Route Types', 2018) to Wikibase items

Route Type	**Description**	**Wikibase item**
100	Railway Service	train service (Q15141321)
101	High Speed Rail Service	
102	Long Distance Trains	
103	Inter Regional Rail Service	
104	Car Transport Rail Service	
105	Sleeper Rail Service	
106	Regional Rail Service	
107	Tourist Railway Service	
108	Rail Shuttle (Within Complex)	
109	Suburban Railway	
110	Replacement Rail Service	
111	Special Rail Service	
112	Lorry Transport Rail Service	
113	All Rail Services	
114	Cross-Country Rail Service	
115	Vehicle Transport Rail Service	
116	Rack and Pinion Railway	
117	Additional Rail Service	
200	Coach Service	public transport bus service (Q1740966)
201	International Coach Service	
202	National Coach Service	
203	Shuttle Coach Service	
204	Regional Coach Service	
205	Special Coach Service	
206	Sightseeing Coach Service	
207	Tourist Coach Service	
208	Commuter Coach Service	
209	All Coach Services	
300	Suburban Railway Service	train service (Q15141321)
400	Urban Railway Service	train service (Q15141321)
401	Metro Service	
402	Underground Service	
403	Urban Railway Service	

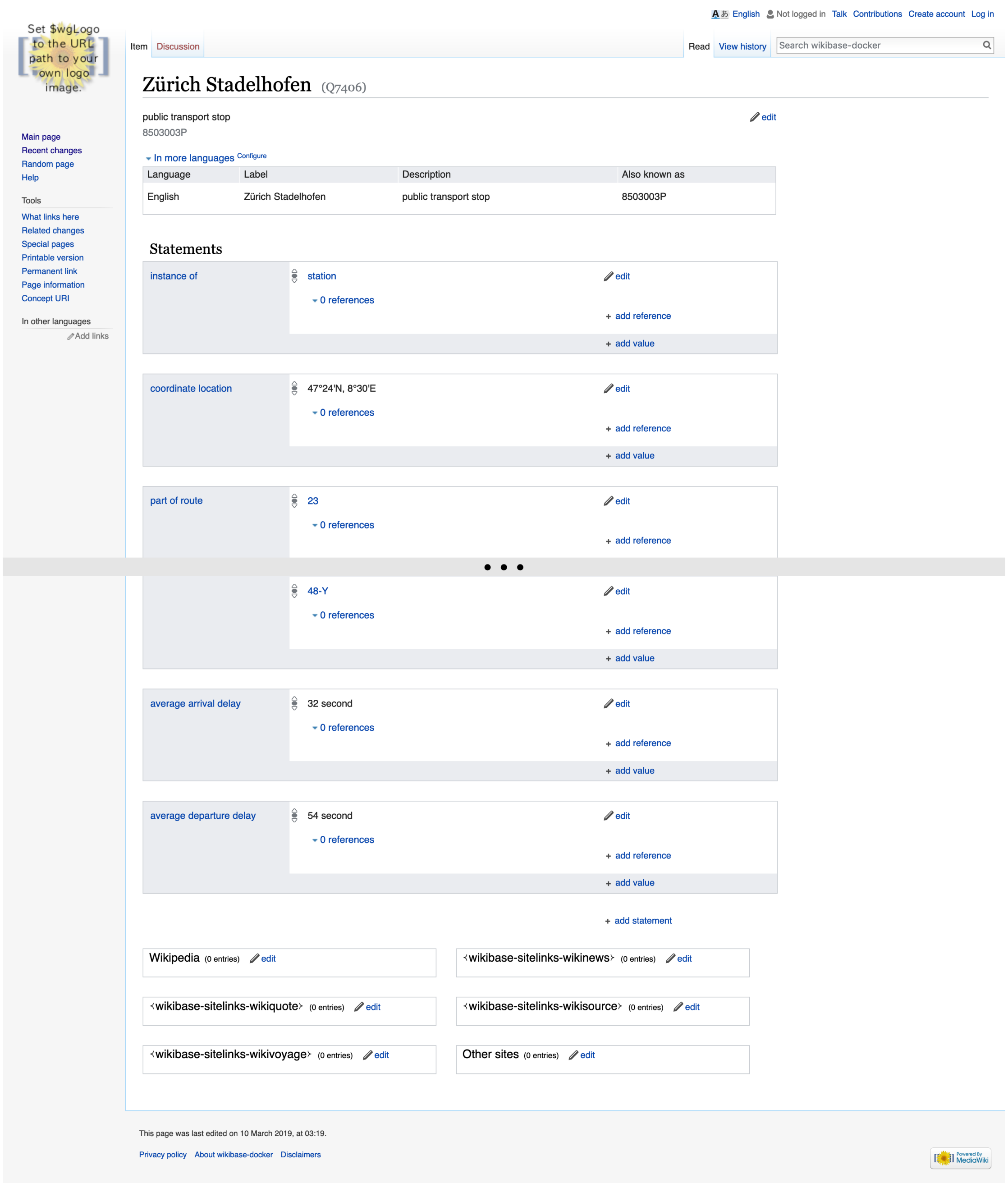
404	All Urban Railway Services	
405	Monorail	
500	Metro Service	rapid transit service (new)
600	Underground Service	rapid transit service (new)
700	Bus Service	public transport bus service (Q1740966)
701	Regional Bus Service	
702	Express Bus Service	
703	Stopping Bus Service	
704	Local Bus Service	
705	Night Bus Service	
706	Post Bus Service	
707	Special Needs Bus	
708	Mobility Bus Service	
709	Mobility Bus for Registered Disabled	
710	Sightseeing Bus	
711	Shuttle Bus	
712	School Bus	
713	School and Public Service Bus	
714	Rail Replacement Bus Service	
715	Demand and Response Bus Service	
716	All Bus Services	
800	Trolleybus Service	public transport bus service (Q1740966)
900	Tram Service	tram service (Q26252263)
901	City Tram Service	
902	Local Tram Service	
903	Regional Tram Service	
904	Sightseeing Tram Service	
905	Shuttle Tram Service	
906	All Tram Services	
1000	Water Transport Service	
1001	International Car Ferry Service	
1002	National Car Ferry Service	

maritime transport (Q155930)

1003	Regional Car Ferry Service	
1004	Local Car Ferry Service	
1005	International Passenger Ferry Service	
1006	National Passenger Ferry Service	
1007	Regional Passenger Ferry Service	
1008	Local Passenger Ferry Service	
1009	Post Boat Service	
1010	Train Ferry Service	
1011	Road-Link Ferry Service	
1012	Airport-Link Ferry Service	
1013	Car High-Speed Ferry Service	
1014	Passenger High-Speed Ferry Service	
1015	Sightseeing Boat Service	
1016	School Boat	
1017	Cable-Drawn Boat Service	
1018	River Bus Service	
1019	Scheduled Ferry Service	
1020	Shuttle Ferry Service	
1021	All Water Transport Services	
1100	Air Service	air transport (Q1757562)
1101	International Air Service	
1102	Domestic Air Service	
1103	Intercontinental Air Service	
1104	Domestic Scheduled Air Service	
1105	Shuttle Air Service	
1106	Intercontinental Charter Air Service	
1107	International Charter Air Service	
1108	Round-Trip Charter Air Service	
1109	Sightseeing Air Service	
1110	Helicopter Air Service	
1111	Domestic Charter Air Service	
1112	Schengen-Area Air Service	
1113	Airship Service	

1114	All Air Services	
1200	Ferry Service	ferry service (new)
1300	Telecabin Service	telecabin service (new)
1301	Telecabin Service	
1302	Cable Car Service	
1303	Elevator Service	
1304	Chair Lift Service	
1305	Drag Lift Service	
1306	Small Telecabin Service	
1307	All Telecabin Services	
1400	Funicular Service	funicular service (Q142031)
1401	Funicular Service	
1402	All Funicular Service	
1500	Taxi Service	taxi service (Q41222493)
1501	Communal Taxi Service	
1502	Water Taxi Service	
1503	Rail Taxi Service	
1504	Bike Taxi Service	
1505	Licensed Taxi Service	
1506	Private Hire Service Vehicle	
1507	All Taxi Services	
1600	Self Drive	self-drive service (new)
1601	Hire Car	
1602	Hire Van	
1603	Hire Motorbike	
1604	Hire Cycle	
1700	Miscellaneous Service	public transport (Q178512)
1701	Cable Car	
1702	Horse-drawn Carriage	

A.3 Wikibase Items



English Not logged in Talk Contributions Create account Log in

Set $wgLogo to the URL path to your own logo image.

Item Discussion Read View history Search wikibase-docker

Main page
Recent changes
Random page
Help

Tools
What links here
Related changes
Special pages
Printable version
Permanent link
Page information
Concept URI

In other languages
Add links

Zürich Stadelhofen (Q7406)

public transport stop
8503003P
edit

In more languages Configure

Language	Label	Description	Also known as
English	Zürich Stadelhofen	public transport stop	8503003P

Statements

Property	Value
instance of	station (0 references)
coordinate location	47°24'N, 8°30'E (0 references)
part of route	23 (0 references)
	• • •
	48-Y (0 references)
average arrival delay	32 second (0 references)
average departure delay	54 second (0 references)

edit · add reference · add value

add statement

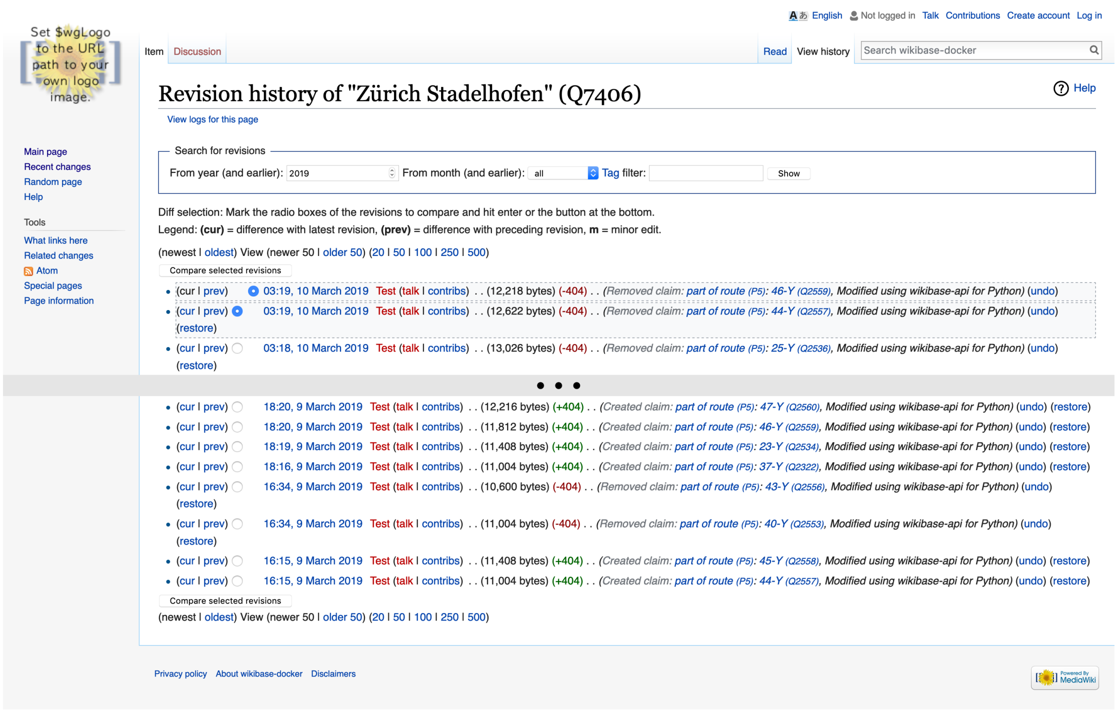
Wikipedia (0 entries) edit
‹wikibase-sitelinks-wikinews› (0 entries) edit
‹wikibase-sitelinks-wikiquote› (0 entries) edit
‹wikibase-sitelinks-wikisource› (0 entries) edit
‹wikibase-sitelinks-wikivoyage› (0 entries) edit
Other sites (0 entries) edit

This page was last edited on 10 March 2019, at 03:19.

Privacy policy About wikibase-docker Disclaimers

Powered By MediaWiki

Figure A.1: Screenshot of an inserted GTFS stop (Zürich Stadelhofen) in Wikibase



Set $wgLogo to the URL path to your own logo image.

Main page
Recent changes
Random page
Help

Tools
What links here
Related changes
Atom
Special pages
Page information

English Not logged in Talk Contributions Create account Log in

Item Discussion | Read View history | Search wikibase-docker

Revision history of "Zürich Stadelhofen" (Q7406)

Help

View logs for this page

Search for revisions
From year (and earlier): 2019 From month (and earlier): all Tag filter: Show

Diff selection: Mark the radio boxes of the revisions to compare and hit enter or the button at the bottom.
Legend: **(cur)** = difference with latest revision, **(prev)** = difference with preceding revision, **m** = minor edit.

(newest | oldest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)

Compare selected revisions

- (cur | prev) 03:19, 10 March 2019 Test (talk | contribs) . . (12,218 bytes) (-404) . . *(Removed claim: part of route (P5): 46-Y (Q2559), Modified using wikibase-api for Python)* (undo)
- (cur | prev) 03:19, 10 March 2019 Test (talk | contribs) . . (12,622 bytes) (-404) . . *(Removed claim: part of route (P5): 44-Y (Q2557), Modified using wikibase-api for Python)* (undo) (restore)
- (cur | prev) 03:18, 10 March 2019 Test (talk | contribs) . . (13,026 bytes) (-404) . . *(Removed claim: part of route (P5): 25-Y (Q2536), Modified using wikibase-api for Python)* (undo) (restore)

• • •

- (cur | prev) 18:20, 9 March 2019 Test (talk | contribs) . . (12,216 bytes) (+404) . . *(Created claim: part of route (P5): 47-Y (Q2560), Modified using wikibase-api for Python)* (undo) (restore)
- (cur | prev) 18:20, 9 March 2019 Test (talk | contribs) . . (11,812 bytes) (+404) . . *(Created claim: part of route (P5): 46-Y (Q2559), Modified using wikibase-api for Python)* (undo) (restore)
- (cur | prev) 18:19, 9 March 2019 Test (talk | contribs) . . (11,408 bytes) (+404) . . *(Created claim: part of route (P5): 23-Y (Q2534), Modified using wikibase-api for Python)* (undo) (restore)
- (cur | prev) 18:16, 9 March 2019 Test (talk | contribs) . . (11,004 bytes) (+404) . . *(Created claim: part of route (P5): 37-Y (Q2322), Modified using wikibase-api for Python)* (undo) (restore)
- (cur | prev) 16:34, 9 March 2019 Test (talk | contribs) . . (10,600 bytes) (-404) . . *(Removed claim: part of route (P5): 43-Y (Q2556), Modified using wikibase-api for Python)* (undo) (restore)
- (cur | prev) 16:34, 9 March 2019 Test (talk | contribs) . . (11,004 bytes) (-404) . . *(Removed claim: part of route (P5): 40-Y (Q2553), Modified using wikibase-api for Python)* (undo) (restore)
- (cur | prev) 16:15, 9 March 2019 Test (talk | contribs) . . (11,408 bytes) (+404) . . *(Created claim: part of route (P5): 45-Y (Q2558), Modified using wikibase-api for Python)* (undo) (restore)
- (cur | prev) 16:15, 9 March 2019 Test (talk | contribs) . . (11,004 bytes) (+404) . . *(Created claim: part of route (P5): 44-Y (Q2557), Modified using wikibase-api for Python)* (undo) (restore)

Compare selected revisions

(newest | oldest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)

Privacy policy About wikibase-docker Disclaimers

Powered By MediaWiki

Figure A.2: Screenshot of the change history of an inserted GTFS stop (Zürich Stadelhofen) in Wikibase

A.4 Dynamics Vocabulary

A.4.1 Turtle Representation

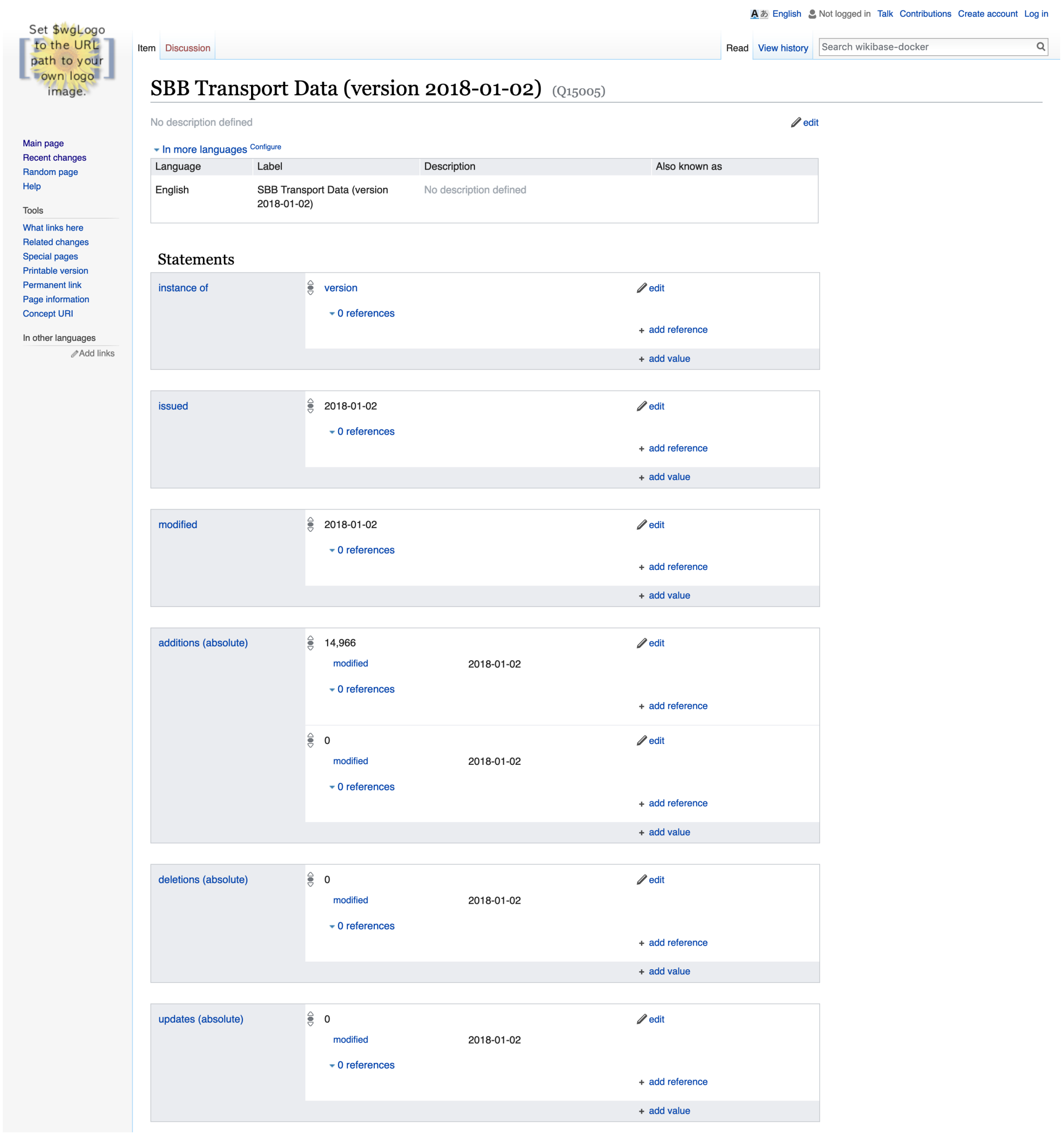
Listing A.4: Extension to the DCAT vocabulary for modelling the dynamics in a LOD dataset, applied to the SBB data

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix dyn: <[DYNAMICS_VOCABULARY_URL]#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .


:uzh a foaf:Organization ;
```

```
  rdfs:label "University of Zurich" ;
  foaf:homepage <https://www.uzh.ch> .

:sbb a foaf:Organization ;
  rdfs:label "Swiss Federal Railways (SBB CFF FFS)" ;
  foaf:homepage <https://www.sbb.ch> .

:agency a dyn:Entity ;
  rdfs:label "Agency" .

:route a dyn:Entity ;
  rdfs:label "Route" .

:stop a dyn:Entity ;
  rdfs:label "Stop" .

:catalog a dcat:Catalog ;
  dct:title "Transport Wiki" ;
  rdfs:label "Transport Wiki" ;
  foaf:homepage <[WIKI_URL]> ;
  dct:subject <http://dbpedia.org/resource/Public_transport> ;
  dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
  dct:publisher :uzh ;
  dcat:dataset :dataset-sbb .

:dataset-sbb a dcat:Dataset ;
  dct:title "SBB Transport Data" ;
  rdfs:label "SBB Transport Data" ;
  dcat:keyword "public transport", "transport", "network", "gtfs", "sbb" ;
  dcat:contactPoint <mailto:data@sbb.ch> ;
  dct:publisher :sbb ;
  dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
  dcat:landingPage <https://opentransportdata.swiss/dataset/timetable
      -2018-gtfs> ;
  dyn:versions :dataset-sbb-versions .

:dataset-sbb-versions a dyn:VersionHistory ;
  dyn:version :dataset-sbb-v1 ;
  dyn:version :dataset-sbb-v2 .


```

```
# Version 1

:dataset-sbb-v1 a dyn:Version ;
  dcat:distribution :gtfsfp20182018-11-21.zip ;
  dct:issued "2018-11-21"^^xsd:date ;
  dct:modified "2018-11-21"^^xsd:date ;
  dyn:entityVersion :agencies-v1 ;
  dyn:entityVersion :routes-v1 ;
  dyn:entityVersion :stops-v1 ;
  dyn:additionsAbs [NR_OF_ADDITIONS] ;
  dyn:additionsRel [PERCENTAGE_ADDITIONS] ;
  dyn:deletionsAbs [NR_OF_DELETIONS] ;
  dyn:deletionsRel [PERCENTAGE_DELETIONS] ;
  dyn:updatesAbs [NR_OF_UPDATES] ;
  dyn:updatesRel [PERCENTAGE_UPDATES] .

:gtfsfp20182018-11-21.zip a dcat:Distribution ;
  dct:title "GTFS_FP2018_2018-11-21.zip" ;
  rdfs:label "GTFS_FP2018_2018-11-21.zip" ;
  dcat:downloadURL <https://opentransportdata.swiss/dataset/b408f747
      -9838-4c05-bb98-10dac3996f17/resource/50066861-ac4e-41ff-9015-5
      fefda985952/download/gtfsfp20182018-11-21.zip> ;
  dcat:mediaType "zip" ;
  dcat:byteSize "86303506"^^xsd:decimal .

:agencies-v1 a dyn:EntityVersion ;
  dyn:entity :agency ;
  dyn:additionsAbs [NR_OF_ADDITIONS] ;
  dyn:additionsRel [PERCENTAGE_ADDITIONS] ;
  dyn:deletionsAbs [NR_OF_DELETIONS] ;
  dyn:deletionsRel [PERCENTAGE_DELETIONS] ;
  dyn:updatesAbs [NR_OF_UPDATES] ;
  dyn:updatesRel [PERCENTAGE_UPDATES] .

:routes-v1 a dyn:EntityVersion ;
  dyn:entity :route ;
  dyn:additionsAbs [NR_OF_ADDITIONS] ;
  dyn:additionsRel [PERCENTAGE_ADDITIONS] ;
  dyn:deletionsAbs [NR_OF_DELETIONS] ;
  dyn:deletionsRel [PERCENTAGE_DELETIONS] ;
  dyn:updatesAbs [NR_OF_UPDATES] ;
  dyn:updatesRel [PERCENTAGE_UPDATES] .

:stops-v1 a dyn:EntityVersion ;
  dyn:entity :stop ;
  dyn:additionsAbs [NR_OF_ADDITIONS] ;
  dyn:additionsRel [PERCENTAGE_ADDITIONS] ;
  dyn:deletionsAbs [NR_OF_DELETIONS] ;
  dyn:deletionsRel [PERCENTAGE_DELETIONS] ;
  dyn:updatesAbs [NR_OF_UPDATES] ;
  dyn:updatesRel [PERCENTAGE_UPDATES] .


# Version 2

:dataset-sbb-v2 a dyn:Version ;
  dcat:distribution :gtfsfp20182018-11-28.zip ;
  dct:issued "2018-11-28"^^xsd:date ;
  dct:modified "2018-11-28"^^xsd:date ;
  dyn:entityVersion :agencies-v2 ;
  dyn:entityVersion :routes-v2 ;
  dyn:entityVersion :stops-v2 ;
  dyn:additionsAbs [NR_OF_ADDITIONS] ;
  dyn:additionsRel [PERCENTAGE_ADDITIONS] ;
  dyn:deletionsAbs [NR_OF_DELETIONS] ;
  dyn:deletionsRel [PERCENTAGE_DELETIONS] ;
  dyn:updatesAbs [NR_OF_UPDATES] ;
  dyn:updatesRel [PERCENTAGE_UPDATES] .

:gtfsfp20182018-11-28.zip a dcat:Distribution ;
  dct:title "GTFS_FP2018_2018-11-28.zip" ;
  rdfs:label "GTFS_FP2018_2018-11-28.zip" ;
  dcat:downloadURL <https://opentransportdata.swiss/dataset/b408f747
      -9838-4c05-bb98-10dac3996f17/resource/827fc81c-ba76-4bb4-a74e-
      c754bbb188dc/download/gtfsfp20182018-11-28.zip> ;
  dcat:mediaType "zip" ;
  dcat:byteSize "86349619"^^xsd:decimal .

:agencies-v2 a dyn:EntityVersion ;
  dyn:entity :agency ;
  dyn:additionsAbs [NR_OF_ADDITIONS] ;
  dyn:additionsRel [PERCENTAGE_ADDITIONS] ;
  dyn:deletionsAbs [NR_OF_DELETIONS] ;
  dyn:deletionsRel [PERCENTAGE_DELETIONS] ;
  dyn:updatesAbs [NR_OF_UPDATES] ;
  dyn:updatesRel [PERCENTAGE_UPDATES] .

:routes-v2 a dyn:EntityVersion ;
  dyn:entity :route ;
  dyn:additionsAbs [NR_OF_ADDITIONS] ;
  dyn:additionsRel [PERCENTAGE_ADDITIONS] ;
  dyn:deletionsAbs [NR_OF_DELETIONS] ;
  dyn:deletionsRel [PERCENTAGE_DELETIONS] ;
  dyn:updatesAbs [NR_OF_UPDATES] ;
  dyn:updatesRel [PERCENTAGE_UPDATES] .

:stops-v2 a dyn:EntityVersion ;
  dyn:entity :stop ;
  dyn:additionsAbs [NR_OF_ADDITIONS] ;
  dyn:additionsRel [PERCENTAGE_ADDITIONS] ;
  dyn:deletionsAbs [NR_OF_DELETIONS] ;
  dyn:deletionsRel [PERCENTAGE_DELETIONS] ;
  dyn:updatesAbs [NR_OF_UPDATES] ;
  dyn:updatesRel [PERCENTAGE_UPDATES] .
```

A.4.2 Wikibase Representation



SBB Transport Data (version 2018-01-02) (Q15005)

No description defined

edit

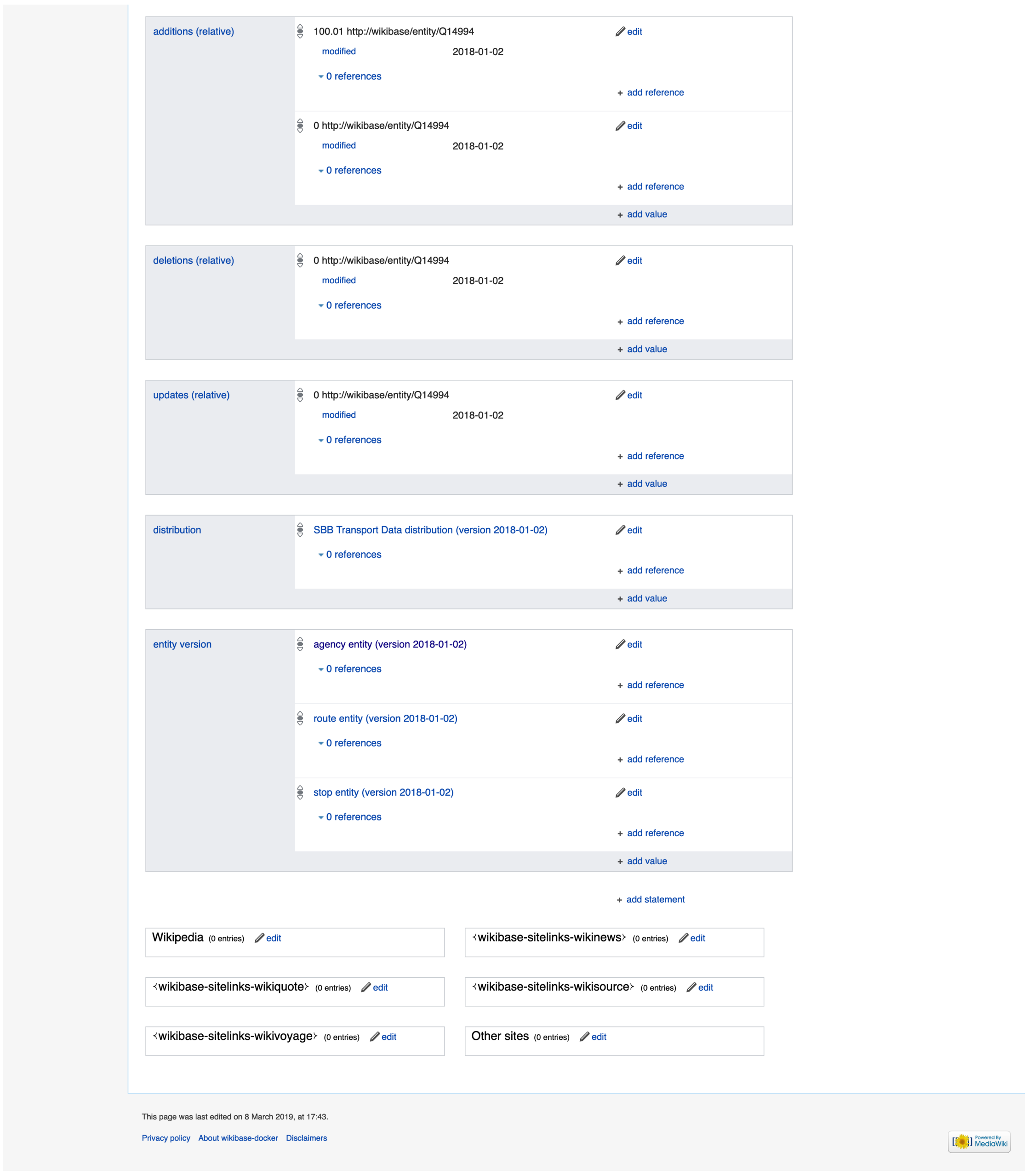
In more languages Configure

Language	Label	Description	Also known as
English	SBB Transport Data (version 2018-01-02)	No description defined	

Statements

instance of	version		edit
	0 references		add reference
			add value
issued	2018-01-02		edit
	0 references		add reference
			add value
modified	2018-01-02		edit
	0 references		add reference
			add value
additions (absolute)	14,966		edit
	modified	2018-01-02	
	0 references		add reference
	0		edit
	modified	2018-01-02	
	0 references		add reference
			add value
deletions (absolute)	0		edit
	modified	2018-01-02	
	0 references		add reference
			add value
updates (absolute)	0		edit
	modified	2018-01-02	
	0 references		add reference
			add value

Figure A.3: Screenshot of a dataset version represented using the dynamics vocabulary in Wikibase (first part)



additions (relative)

100.01 http://wikibase/entity/Q14994 — edit
modified 2018-01-02
0 references
+ add reference

0 http://wikibase/entity/Q14994 — edit
modified 2018-01-02
0 references
+ add reference
+ add value

deletions (relative)

0 http://wikibase/entity/Q14994 — edit
modified 2018-01-02
0 references
+ add reference
+ add value

updates (relative)

0 http://wikibase/entity/Q14994 — edit
modified 2018-01-02
0 references
+ add reference
+ add value

distribution

SBB Transport Data distribution (version 2018-01-02) — edit
0 references
+ add reference
+ add value

entity version

agency entity (version 2018-01-02) — edit
0 references
+ add reference

route entity (version 2018-01-02) — edit
0 references
+ add reference

stop entity (version 2018-01-02) — edit
0 references
+ add reference
+ add value

+ add statement

Wikipedia (0 entries) edit

‹wikibase-sitelinks-wikinews› (0 entries) edit

‹wikibase-sitelinks-wikiquote› (0 entries) edit

‹wikibase-sitelinks-wikisource› (0 entries) edit

‹wikibase-sitelinks-wikivoyage› (0 entries) edit

Other sites (0 entries) edit

This page was last edited on 8 March 2019, at 17:43.

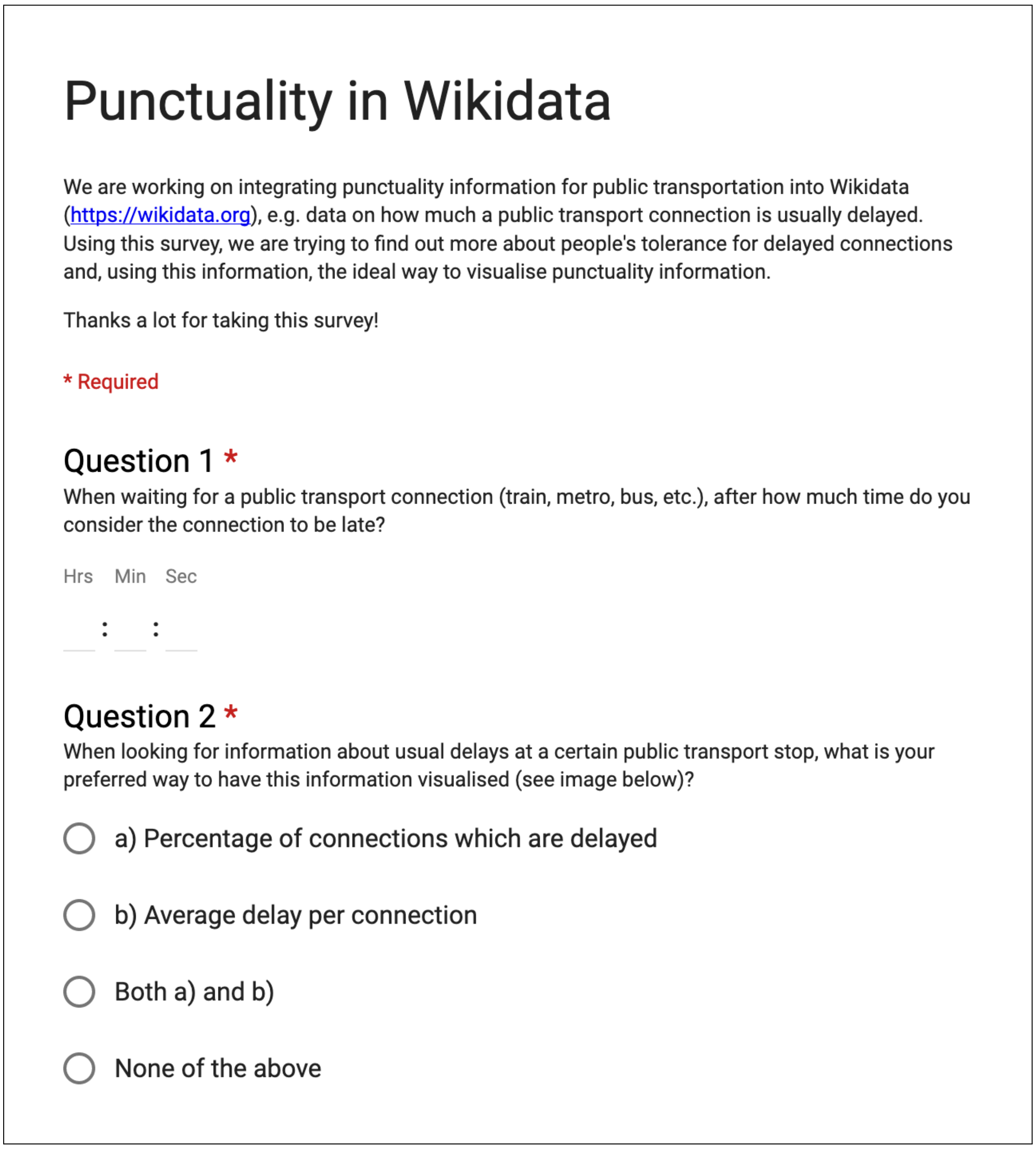
Privacy policy About wikibase-docker Disclaimers

Powered By MediaWiki

Figure A.4: Screenshot of a dataset version represented using the dynamics vocabulary in Wikibase (second part)

A.5 Survey

A.5.1 Survey Questions



Punctuality in Wikidata

We are working on integrating punctuality information for public transportation into Wikidata (https://wikidata.org), e.g. data on how much a public transport connection is usually delayed. Using this survey, we are trying to find out more about people's tolerance for delayed connections and, using this information, the ideal way to visualise punctuality information.

Thanks a lot for taking this survey!

* Required

Question 1 *

When waiting for a public transport connection (train, metro, bus, etc.), after how much time do you consider the connection to be late?

Hrs Min Sec

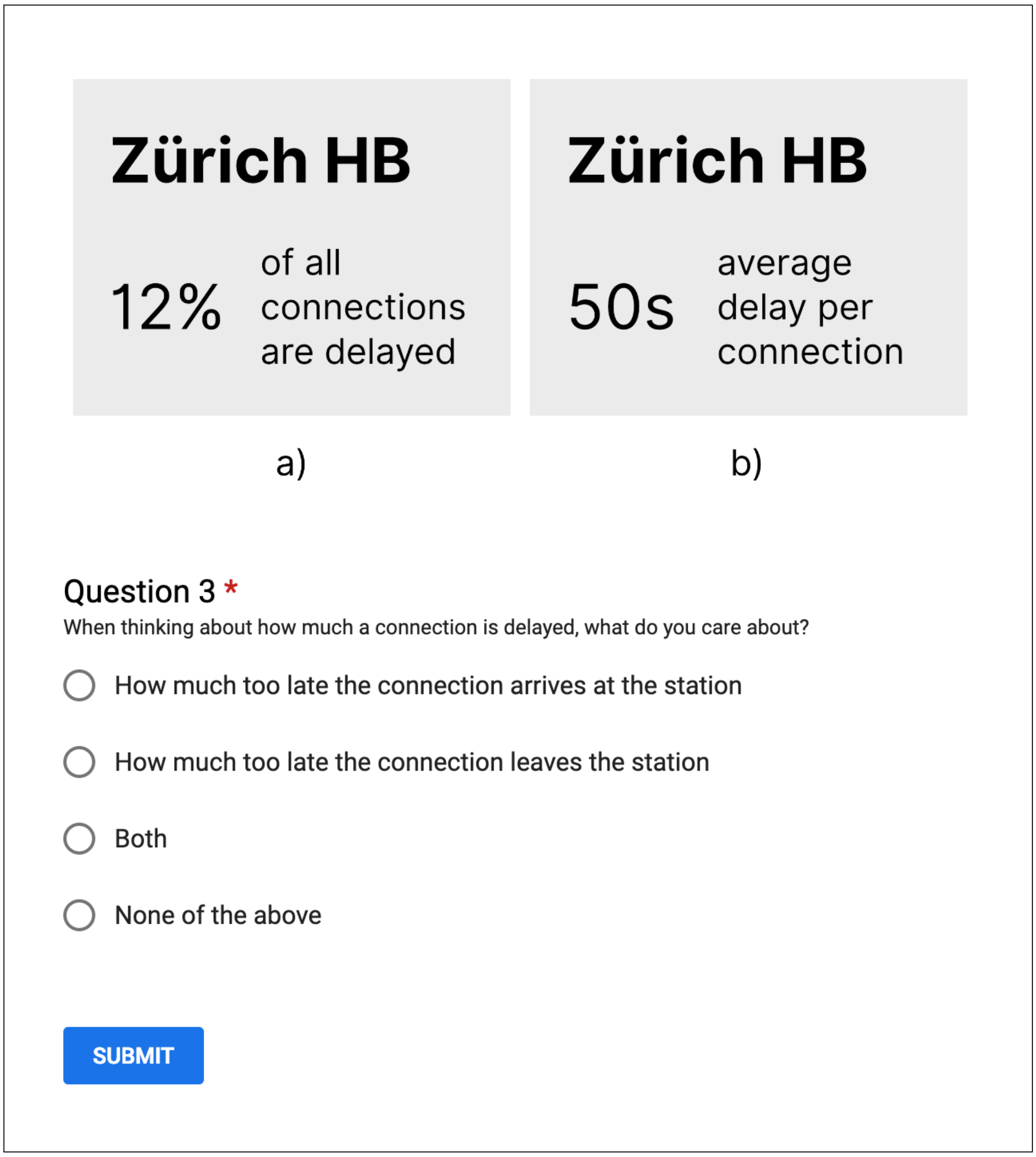
__ : __ : __

Question 2 *

When looking for information about usual delays at a certain public transport stop, what is your preferred way to have this information visualised (see image below)?

- a) Percentage of connections which are delayed
- b) Average delay per connection
- Both a) and b)
- None of the above

Figure A.5: Survey on punctuality in public transport (first part)



Zürich HB

12% of all connections are delayed

a)

Zürich HB

50s average delay per connection

b)

Question 3 *

When thinking about how much a connection is delayed, what do you care about?

○ How much too late the connection arrives at the station

○ How much too late the connection leaves the station

○ Both

○ None of the above

SUBMIT

Figure A.6: Survey on punctuality in public transport (second part)

A.5.2 Survey Results

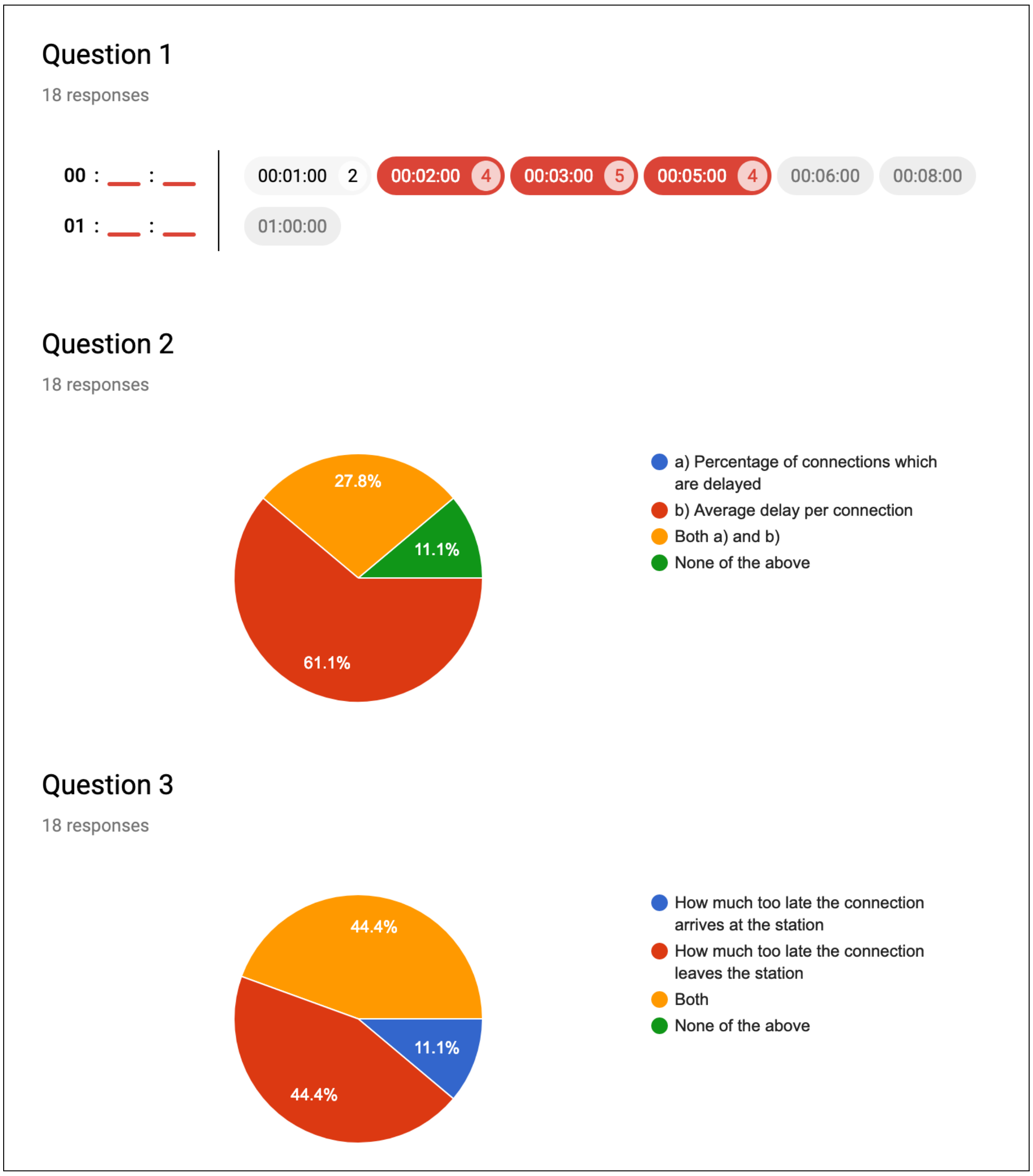


Figure A.7: Results of the survey on punctuality

A.6 Software

All four software libraries/scripts developed for this thesis can be found on the attached CD-ROM. The tools are thoroughly documented. Each project directory contains a `README` file with the documentation or instructions on where to find or how to generate it.

To reproduce the insertions which were analysed in chapter 6, follow these instructions (tested on macOS and Debian):

1. Install Docker CE (*https://docs.docker.com/install*)
2. Install GNU Make (*https://www.gnu.org/software/make*)
3. Clone the `sbb-wikibase` repository with its submodules (`--recurse-submodules` flag)
4. Follow the instructions in the `sbb-wikibase` `README` file and run the program on the SBB's GTFS and 'actual data' datasets from the period of interest:
    - GTFS datasets 2018: *https://opentransportdata.swiss/de/dataset/timetable-2018-gtfs*
    - GTFS datasets 2019: *https://opentransportdata.swiss/de/dataset/timetable-2019-gtfs*
    - 'Actual data' datasets: *https://opentransportdata.swiss/de/dataset/istdaten*

A.7 Data Analysis

Update Statistics

In [1]:

```python
import json
import os
from datetime import datetime
from pathlib import Path

import matplotlib.dates as mdates
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

pd.plotting.register_matplotlib_converters()

CURRENT_DIR = str(Path().resolve())
OUTPUT_DIR = os.path.join(CURRENT_DIR, "output")
UPDATE_STATS_PATH = os.path.join(OUTPUT_DIR, "gtfs", "gtfs-stats.json")
SURVEY_RESULTS_PATH = os.path.join(CURRENT_DIR, "survey-results.csv")
PUNCTUALITY_AVGS_PATH = os.path.join(
    OUTPUT_DIR, "punctuality", "punctuality-avgs.csv"
)
PUNCTUALITY_STATS_PATH = os.path.join(
    OUTPUT_DIR, "punctuality", "punctuality-stats.json"
)
```

Plotting helper functions:

In [2]:

```python
def draw_month_ticks():
    locator = mdates.MonthLocator()
    fmt = mdates.DateFormatter("%b")
    x_axis = plt.gca().xaxis
    x_axis.set_major_locator(locator)
    x_axis.set_major_formatter(fmt)


def draw_timetable_changes():
    # Start of summer schedule
    plt.axvline(datetime(2018, 6, 30), linestyle="dashed", color="black")
    # End of summer schedule
    plt.axvline(datetime(2018, 8, 26), linestyle="dashed", color="black")


def get_monthly_avg(df, col_name):
    return df.groupby(pd.Grouper(freq="M")).aggregate({col_name: np.mean})
```

Visualizations of Insertions/Updates

The vertical lines in the plots below correspond to the following dates:

- 30/06/2018: Start of summer schedule (during which an increased amount of construction work took place)
- 26/08/2018: End of summer schedule
- 09/12/2019: Start of schedule for 2019

Data Preparation

Data parsing and conversion:

In [3]:

```
# Read stats file into DataFrame
if not os.path.isfile(UPDATE_STATS_PATH):
    raise Exception(f"Update stats file is missing at path {UPDATE_STATS_PATH}")
with open(UPDATE_STATS_PATH) as file:
    gtfs_stats = json.load(file)
    df_gtfs = pd.DataFrame.from_dict(gtfs_stats, orient="index")

# Check if there are enough rows
nr_of_updates = len(df_gtfs.index)
if nr_of_updates < 3:
    raise Exception(
        "At least the initial insertion and two updates must have been "
        "performed for the data analysis to be meaningful"
    )

# Delete first row (initial insertion)
df_gtfs = df_gtfs.iloc[1:]

# Parse GTFS update dates
df_gtfs["date_str"] = df_gtfs.index
df_gtfs.index = df_gtfs.apply(
    lambda row: datetime.strptime(row["date_str"], "%Y-%m-%d"), axis=1
)
```

Changes over Item Types

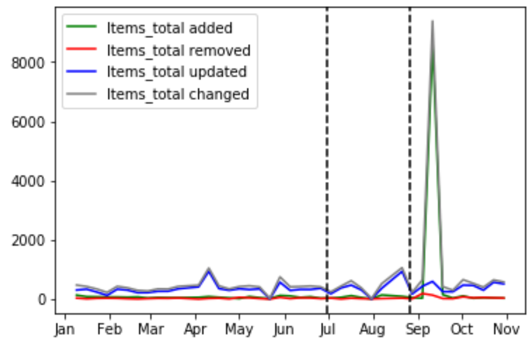
In [4]:

```
def draw_item_type_plot(item_type):
    for change_type in [
        {"label": "added", "color": "green"},
        {"label": "removed", "color": "red"},
        {"label": "updated", "color": "blue"},
    ]:
        plt.plot(
            df_gtfs[f'{item_type}_{change_type["label"]}'],
            color=change_type["color"],
            label=f'{item_type.capitalize()} {change_type["label"]}',
        )
    plt.plot(
        df_gtfs[item_type + "_added"]
        + df_gtfs[item_type + "_removed"]
        + df_gtfs[item_type + "_updated"],
        color="grey",
        label=item_type.capitalize() + " changed",
    )
    draw_timetable_changes()
    plt.legend()
    draw_month_ticks()
    plt.show()
```

Total number of item changes (number of added/removed/updated items):

In [5]:

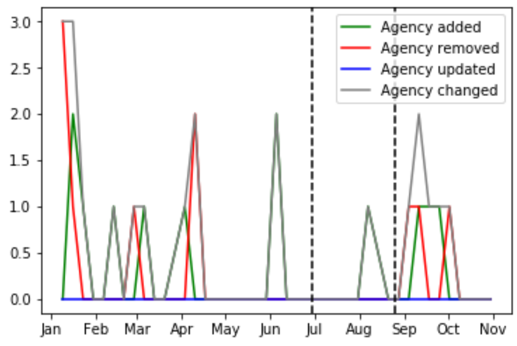
```
draw_item_type_plot("items_total")
```



Total number of agency changes (number of added/removed/updated agency items):

In [6]:

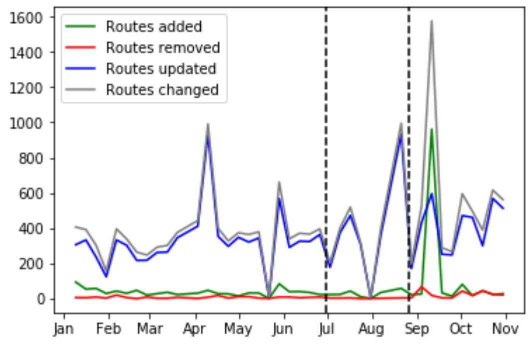
```
draw_item_type_plot("agency")
```



Total number of route changes (number of added/removed/updated route items):

In [7]:

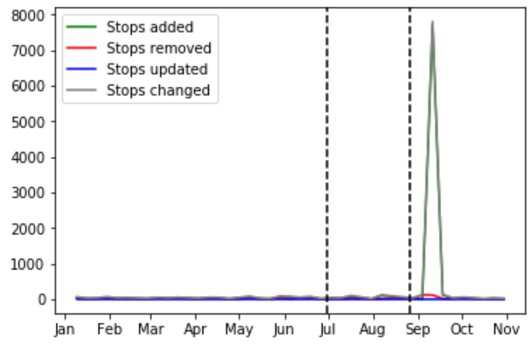
```
draw_item_type_plot("routes")
```



Total number of stop changes (number of added/removed/updated stop items):

In [8]:

```
draw_item_type_plot("stops")
```

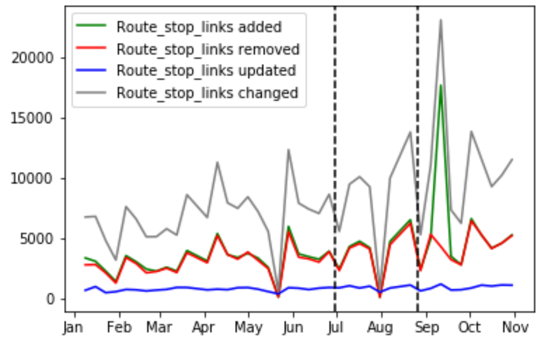


Total number of changes (number of added/removed/updated stops):

Note: A single change in how stops are traversed will result in multiple claim changes. For instance, if a stop is no longer part of a route, both the stop's `part of route` claim as well as the route's `serves station` claim (including the `stop index` qualifier) route are removed. As another example, if a stop's stop index changes, the indices of other stops on the route might change as well. Furthermore, if new routes or stops are created, multiple `serves station` / `part of route` claims will be added as well.

In [9]:

```
draw_item_type_plot("route_stop_links")
```



Change Distribution

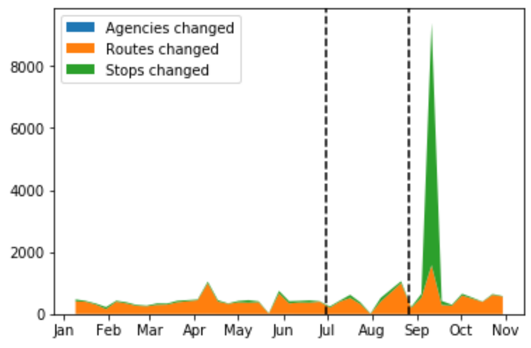
In [10]:

```
def draw_change_type_plot(change_type):
    plt.stackplot(
        df_gtfs.index,
        df_gtfs[f"agency_{change_type}"],
        df_gtfs[f"routes_{change_type}"],
        df_gtfs[f"stops_{change_type}"],
        labels=[
            f"Agencies {change_type}",
            f"Routes {change_type}",
            f"Stops {change_type}",
        ],
    )
    draw_timetable_changes()
    plt.legend()
    draw_month_ticks()
    plt.show()
```

Distribution of all changes over entity types:

In [11]:

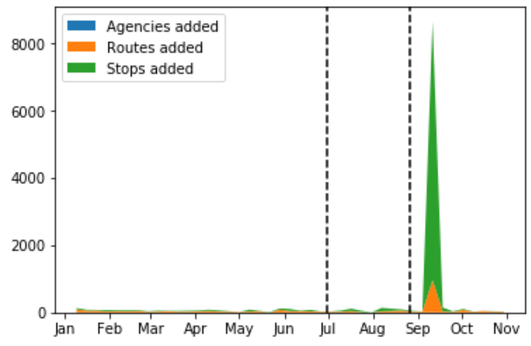
```
plt.stackplot(
    df_gtfs.index,
    df_gtfs["agency_added"]
    + df_gtfs["agency_removed"]
    + df_gtfs["agency_updated"],
    df_gtfs["routes_added"]
    + df_gtfs["routes_removed"]
    + df_gtfs["routes_updated"],
    df_gtfs["stops_added"]
    + df_gtfs["stops_removed"]
    + df_gtfs["stops_updated"],
    labels=["Agencies changed", "Routes changed", "Stops changed"],
)
draw_timetable_changes()
plt.legend()
draw_month_ticks()
plt.show()
```



Distribution of additions over entity types:

In [12]:

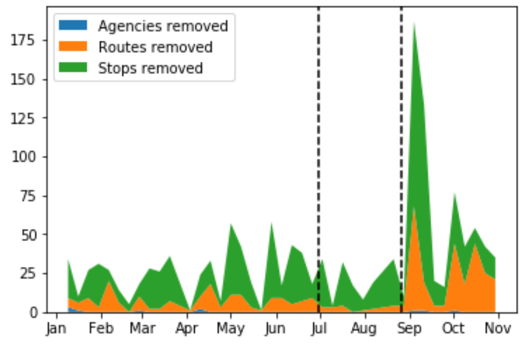
```
draw_change_type_plot("added")
```



Distribution of deletions over entity types:

In [13]:

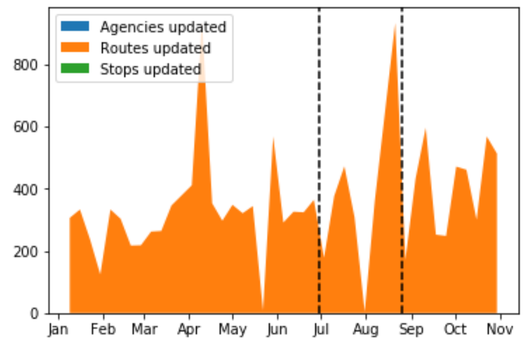
```
draw_change_type_plot("removed")
```



Distribution of updates over entity types:

In [14]:

```
draw_change_type_plot("updated")
```



Punctuality

Survey Results

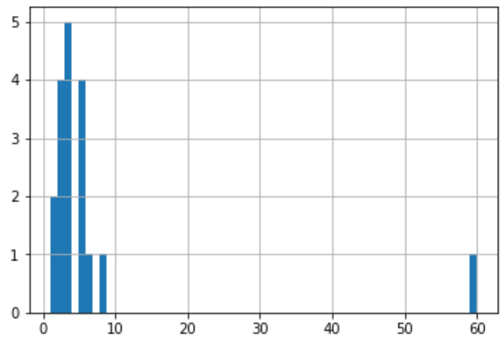
In [15]:

```
ftr = [3600, 60, 1]


def time_str_to_s(time_str):
    return sum([a * b for a, b in zip(ftr, map(int, time_str.split(":")))])


if not os.path.isfile(SURVEY_RESULTS_PATH):
    raise Exception(
        f"Survey results file is missing at path {SURVEY_RESULTS_PATH}"
    )
df_survey = pd.read_csv(SURVEY_RESULTS_PATH)

# Convert timestamps to seconds
df_survey["q1_seconds"] = df_survey.apply(
    lambda row: time_str_to_s(row["Question 1"]), axis=1
)
```

Question 1:

In [16]:

```
df_survey["q1_minutes"] = df_survey["q1_seconds"] / 60
df_survey["q1_minutes"].hist(bins=60)
plt.show()
```



In [17]:

```
df_survey["q1_minutes"].describe()
```

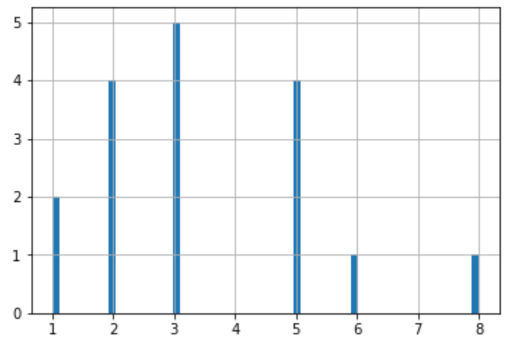
Out[17]:

```
count    18.000000
mean      6.611111
std      13.452045
min       1.000000
25%       2.000000
50%       3.000000
75%       5.000000
max      60.000000
Name: q1_minutes, dtype: float64
```

Without outliers (removed using interquartile range):

In [18]:

```
q1 = df_survey["q1_minutes"].quantile(0.25)
q3 = df_survey["q1_minutes"].quantile(0.75)
iqr = q3 - q1
lower_bound = q1 - (1.5 * iqr)
upper_bound = q3 + (1.5 * iqr)

df_survey.loc[
    (df_survey["q1_minutes"] > lower_bound)
    & (df_survey["q1_minutes"] < upper_bound)
]["q1_minutes"].hist(bins=60)
plt.show()
```



In [19]:

```
df_survey.loc[
    (df_survey["q1_minutes"] > lower_bound)
    & (df_survey["q1_minutes"] < upper_bound)
]["q1_minutes"].describe()
```

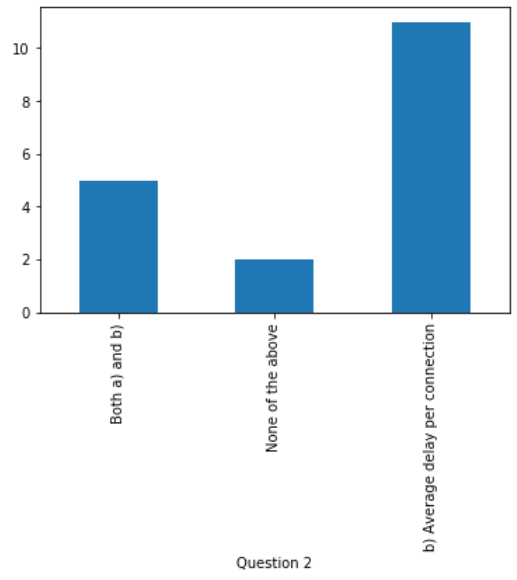
Out[19]:

```
count    17.000000
mean      3.470588
std       1.907801
min       1.000000
25%       2.000000
50%       3.000000
75%       5.000000
max       8.000000
Name: q1_minutes, dtype: float64
```

Question 2:

In [20]:

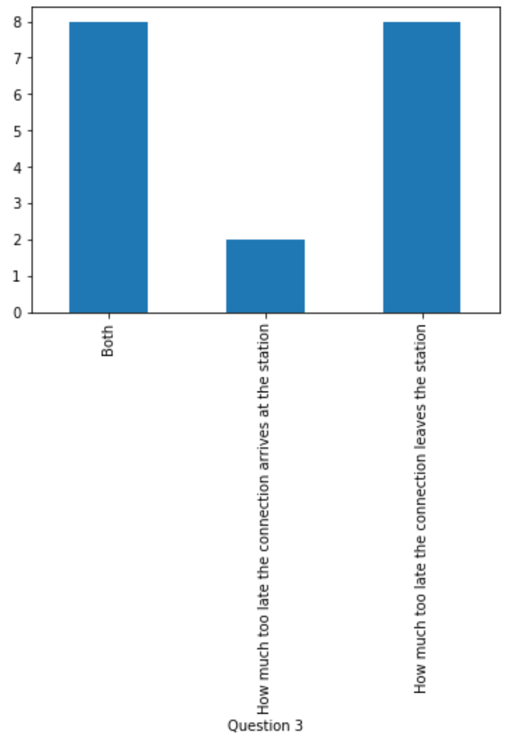
```
df_survey["Question 2"].groupby(df_survey["Question 2"]).count().plot(
    kind="bar"
)
plt.show()
```



Question 3:

```
In [21]:
```

```
df_survey["Question 3"].groupby(df_survey["Question 3"]).count().plot(
    kind="bar"
)
plt.show()
```



Average Delay Over Time

In [22]:

```
if not os.path.isfile(PUNCTUALITY_STATS_PATH):
    raise Exception(
        f"Punctuality stats file is missing at path {PUNCTUALITY_STATS_PATH}"
    )
with open(PUNCTUALITY_STATS_PATH) as file:
    pct_stats = json.load(file)
    df_pct_stats = pd.DataFrame.from_dict(pct_stats, orient="index")

# Parse punctuality stats dates
df_pct_stats["date_str"] = df_pct_stats.index
df_pct_stats.index = df_pct_stats.apply(
    lambda row: datetime.strptime(row["date_str"], "%Y-%m-%d"), axis=1
)
```

In [23]:

```
delays_filtered = df_pct_stats[
    (df_pct_stats["avg_arr_delay"]) > 0 & (df_pct_stats["avg_dep_delay"] > 0)
]
```

Average daily delay per station:

Arrival delay:

In [24]:

```
delays_filtered["avg_arr_delay"].describe()
```

Out[24]:

```
count    180.000000
mean      83.092274
std       13.848601
min       15.875004
25%       75.408758
50%       82.889635
75%       89.022895
max      131.623062
Name: avg_arr_delay, dtype: float64
```

Departure delay:

In [25]:

```
delays_filtered["avg_dep_delay"].describe()
```

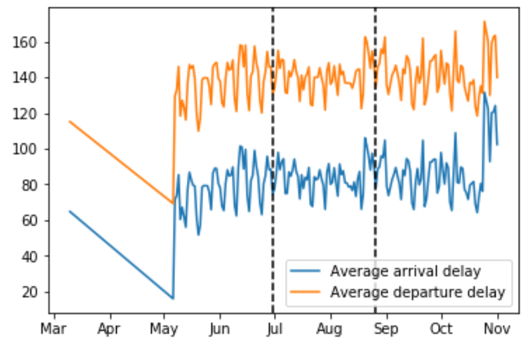
Out[25]:

```
count     180.000000
mean      139.727203
std        12.498760
min        69.249997
25%       132.701493
50%       140.080060
75%       146.539571
max       171.361437
Name: avg_dep_delay, dtype: float64
```

Average daily delay over time:

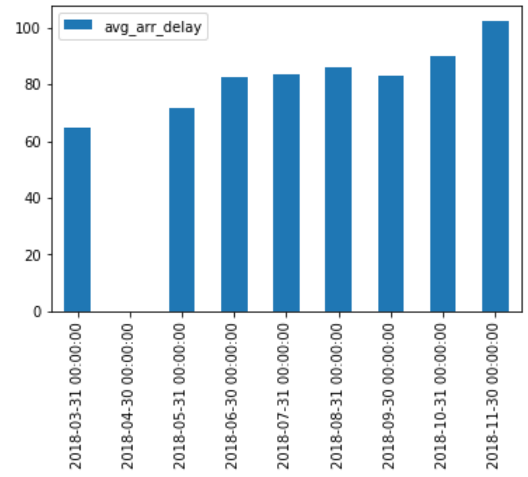
In [26]:

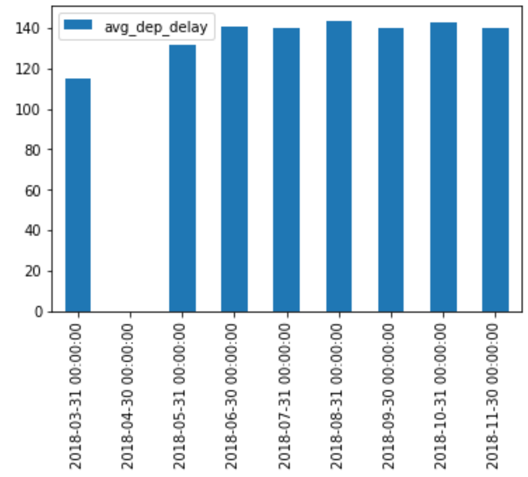
```
plt.plot(delays_filtered["avg_arr_delay"], label="Average arrival delay")
plt.plot(delays_filtered["avg_dep_delay"], label="Average departure delay")
draw_timetable_changes()
plt.legend()
draw_month_ticks()
plt.show()
```



In [27]:

```
get_monthly_avg(delays_filtered, "avg_arr_delay").plot.bar()
get_monthly_avg(delays_filtered, "avg_dep_delay").plot.bar()
plt.show()
```





Average Delay for Previous Day

In [28]:

```
if not os.path.isfile(PUNCTUALITY_AVGS_PATH):
    raise Exception(
        f"Punctuality averages file is missing at path {PUNCTUALITY_AVGS_PATH}"
    )
df_pct_avg = pd.read_csv(PUNCTUALITY_AVGS_PATH)
```

Descriptive statistics on arrival delay for previous day:

In [29]:

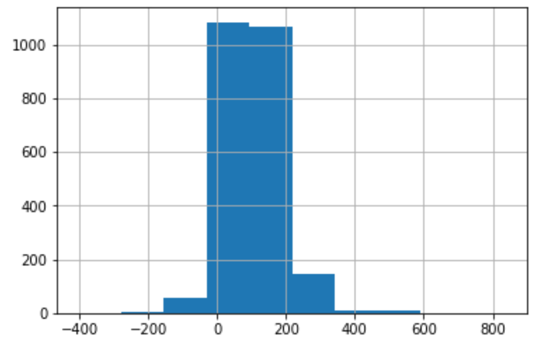
```
df_pct_avg["avg_arrival_delay"].describe()
```

Out[29]:

```
count    2381.000000
mean      102.401108
std        77.846261
min      -401.999999
25%        59.904763
50%        95.999997
75%       138.175997
max       838.000001
Name: avg_arrival_delay, dtype: float64
```

In [30]:

```
df_pct_avg["avg_arrival_delay"].hist()
plt.show()
```



Descriptive statistics on departure delay for previous day:

In [31]:

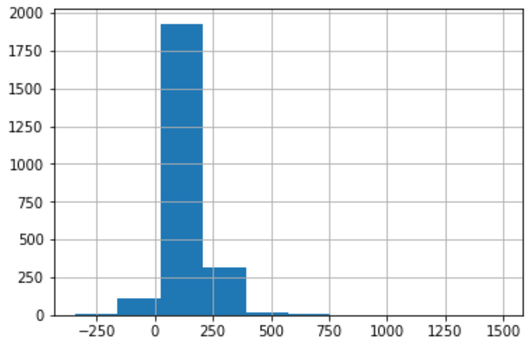
```
df_pct_avg["avg_departure_delay"].describe()
```

Out[31]:

```
count    2381.000000
mean      139.978315
std        86.640789
min      -343.999992
25%        92.714287
50%       131.186667
75%       175.000001
max      1493.999983
Name: avg_departure_delay, dtype: float64
```

In [32]:

```
df_pct_avg["avg_departure_delay"].hist()
plt.show()
```



Descriptive statistics on difference between arrival and departure delay for previous day:

While at a stop, the delay increased by…

In [33]:

```
df_pct_avg["avg_difference"] = (
    df_pct_avg["avg_departure_delay"] - df_pct_avg["avg_arrival_delay"]
)
```

In [34]:

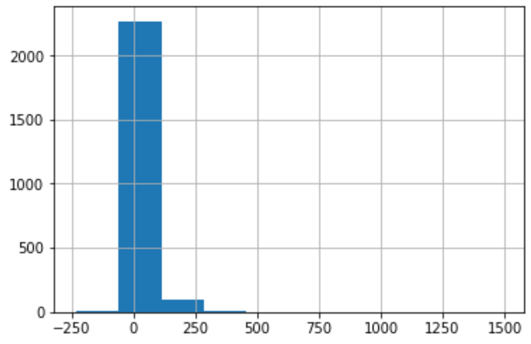
```
df_pct_avg["avg_difference"].describe()
```

Out[34]:

```
count    2381.000000
mean       37.577207
std        55.610384
min      -232.875622
25%        15.666676
50%        27.866664
75%        45.379311
max      1493.999983
Name: avg_difference, dtype: float64
```

In [35]:

```
df_pct_avg["avg_difference"].hist()
plt.show()
```



Stops where delay increased by more than a minute between arrival and departure for previous day:

```
In [36]:
df_pct_avg.loc[df_pct_avg["avg_difference"] < -60].sort_values(
    "avg_difference", ascending=False
)
Out[36]:
```

	stop_id	avg_arrival_delay	avg_departure_delay	stop_id.1	stop_name	stop_desc
1183	8575221	242.363628	174.181824	8575221	Biasca, Stazione	NaN
961	8508261	241.013889	172.569447	8508261	Biglen	NaN
1546	8575700	-22.000015	-93.999986	8575700	Maroggia-Melano, Stazione	NaN
964	8508264	256.227275	180.340913	8508264	Schafhausen i.E.	NaN
885	8507372	94.761904	16.333335	8507372	Wengen	NaN
954	8508251	257.358975	173.179488	8508251	Steffisburg	NaN
960	8508260	262.681817	176.878787	8508260	Grosshöchstetten	NaN
957	8508254	225.833336	130.250001	8508254	Brenzikofen	NaN
1418	8575526	180.181817	64.727269	8575526	Magadino, Debarcadero	NaN
1904	8580788	216.909103	84.909088	8580788	Cadenazzo, Stazione	NaN
2209	8591603	62.656716	-170.218906	8591603	Lugano, Centro	NaN

References

Alexander, K., Cyganiak, R., Hausenblas, M. & Zhao, J. (2011, March 3). Describing Linked Datasets with the VoID Vocabulary. Retrieved November 23, 2018, from *https://www.w3.org/TR/void/*

Antrim, A. & Barbeau, S. J. (2013, April). The Many Uses of GTFS Data – Opening the Door to Transit and Multimodal Applications. In *ITS America's 23rd Annual Meeting & Exposition*. Retrieved January 10, 2019, from *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.391.5421*

Ayers, P., Matthews, C. & Yates, B. (2008, September 16). *How Wikipedia Works: And How You Can Be a Part of It*. No Starch Press. Retrieved January 9, 2019, from *https://archive.org/details/HowWikipediaWorks/*

Berners-Lee, T. (2000, November 7). *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web* (1st edition). San Francisco: HarperBusiness.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001, May). The Semantic Web. *Scientific American*, *284*(5), 28–37. c.

Brickley, D. & Miller, L. (2014, January 14). FOAF Vocabulary Specification 0.99. Retrieved January 8, 2019, from *http://xmlns.com/foaf/spec/*

Cyganiak, R., Wood, D., Lanthaler, M., Klyne, G., Carroll, J. J. & McBride, B. (2014, February 25). RDF 1.1 Concepts and Abstract Syntax. Retrieved January 7, 2019, from *https://www.w3.org/TR/rdf11-concepts/*

Davis, M. (2011, September 27). Data model diagrams for GTFS. Retrieved January 2, 2019, from *https://lin-ear-th-inking.blogspot.com/2011/09/data-model-diagrams-for-gtfs.html*

Dividino, R., Gottron, T., Scherp, A. & Gröner, G. (2014). From Changes to Dynamics: Dynamics Analysis of Linked Open Data Sources.

Erxleben, F., Günther, M., Krötzsch, M., Mendez, J. & Vrandečić, D. (2014). Introducing Wikidata to the Linked Data Web. In *The Semantic Web – ISWC 2014* (pp. 50–65). doi:*10.1007/978-3-319-11964-9_4*

Extended GTFS Route Types. (2018, December 11). Retrieved January 3, 2019, from *https://developers.google.com/transit/gtfs/reference/extended-route-types*

General Transit Feed Specification Reference. (2018, October 18). Retrieved January 2, 2019, from *https://developers.google.com/transit/gtfs/reference/*

GTFS Static Overview. (2016, July 26). Retrieved September 5, 2018, from *https://developers.google.com/transit/gtfs*

Gutierrez, C., Hurtado, C. & Vaisman, A. (2007, February). Introducing Time into RDF. *IEEE Transactions on Knowledge and Data Engineering*, *19*(2), 207–218. doi:*10.1109/TKDE.2007.34*

Hayes, P. J. & Patel-Schneider, P. F. (2014, February 25). RDF 1.1 Semantics. Retrieved January 8, 2019, from *https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/*

Help:Statements. (2018, December 27). Retrieved March 2, 2019, from *https://www.wikidata.org/wiki/Help:Statements*

Krötzsch, M. & Vrandečić, D. (2014, October). Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, *57*(10), 78–85. doi:*10.1145/2629489*

List of Wikipedias. (2018, December 25). Retrieved January 9, 2019, from *https://meta.wikimedia.org/wiki/List_of_Wikipedias*

Maali, F., DERI, NUI Galway, Erickson, J. & Tetherless World Constellation (RPI). (2014, January 16). Data Catalog Vocabulary (DCAT). Retrieved November 22, 2018, from *https://www.w3.org/TR/vocab-dcat/*

Meier, B. & Osterwald, S. (2018, March 22). SBB Facts and Figures 2017. SBB AG. Retrieved August 21, 2018, from *https://reporting.sbb.ch*

Tappolet, J. & Bernstein, A. (2009). Applied Temporal RDF: Efficient Temporal Querying of RDF Data with SPARQL. In *The Semantic Web: Research and Applications* (pp. 308–322). Lecture Notes in Computer Science. Springer Berlin Heidelberg.

Thalhammer, A. (2017). Linked Data Entity Summarization. doi:*10.5445/IR/1000065395*

Vrandečić, D. (2013, July). The Rise of Wikidata. *IEEE Intelligent Systems*, *28*(4), 90–95. doi:*10.1109/MIS.2013.119*

W3C SPARQL Working Group. (2013, March 21). SPARQL 1.1 Overview. Retrieved March 13, 2019, from *https://www.w3.org/TR/sparql11-overview/*

Wikibase/DataModel. (2018, December 12). Retrieved January 9, 2019, from *https://www.mediawiki.org/wiki/Wikibase/DataModel*

Wikibase/Indexing/RDF Dump Format. (2019, January 25). Retrieved March 10, 2019, from *https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format*

Wikidata:Data Access. (2019, February 2). Retrieved March 3, 2019, from *https://www.wikidata.org/wiki/Wikidata:Data_access*

Wikidata:Relation between Properties in RDF and in Wikidata. (2017, July 11). Retrieved January 2, 2019, from *https://www.wikidata.org/wiki/Wikidata:Relation_between_properties_in_RDF_and_in_Wikidata*