Generierung angepasster RDF-Dumps von Wikidata

Benno Fünfstück

June 21, 2016
Version: My First Draft

Technische Universität Dresden

Fakultät Informatik
Institut für Theoretische Informatik
Professur für Wissensbasierte Systeme

Bachelorarbeit

Generierung angepasster RDF-Dumps von Wikidata

Benno Fünfstück

1. Reviewer Prof. Markus Kroetzsch
Fakultät Informatik
Technische Universität Dresden

2. Reviewer Prof. Sebastian Rudolph
Informatik
Technische Universität Dresden

Supervisors Prof. Markus Kroetzsch and Prof. Sebastian Rudolph

June 21, 2016

Benno Fünfstück
Generierung angepasster RDF-Dumps von Wikidata
Bachelorarbeit, June 21, 2016
Reviewers: Prof. Markus Kroetzsch and Prof. Sebastian Rudolph
Supervisors: Prof. Markus Kroetzsch and Prof. Sebastian Rudolph

Technische Universität Dresden
Professur für Wissensbasierte Systeme
Institut für Theoretische Informatik
Fakultät Informatik

01062 and Dresden

Abstract

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: Dies ist ein Blindtext oder Huardest gefburn? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie Lorem ipsum dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Abstract (different language)

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: Dies ist ein Blindtext oder Huardest gefburn? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie Lorem ipsum dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Inhaltsverzeichnis

1 Einleitung

2 Hintergrund

2.1 Wikidata

[VK14]

2.1.1 Wissensgraphen

2.1.2 Datenmodell

2.2 RDF

[@Pro19b]

3 Anforderungen

3.1 funktionale Anforderungen

Im Fokus dieser Arbeit steht die Entwicklung eines Systems zur Filterung des RDF-Exports von Wikidata. Damit sollen Wissenschaftler mit wenig Aufwand Dumps nach speziellen Kriterien erstellen können. Die wesentlichen Merkmale des Systems sind:

Format Der Dump soll als RDF im N-Triples Format erstellt werden. Damit die gefilterten Dumps möglichst kompatibel mit dem vollständigen Wikidata RDF Dump sind, sollte das Schema dem offiziellen RDF-Dump-Format entsprechen. Erweiterungen des Schemas müssen klar gekennzeichnet sein.

Filterung Nutzer sollen über eine einfache Oberfläche wählen können, welche Entitäten in welchem Detailgrad im erstellten Dump enthalten sind.

Archivierung Damit die Dumps in wissenschaftlichen Veröffentlichungen zitiert werden können, muss die Verfügbarkeit auch in ferner Zukunft garantiert werden. Deshalb ist eine Methode zur Langzeitarchivierung generierter Dumps notwendig.

Suche es soll eine durchsuchbare Übersicht über alle erstellen Dumps. So können Ideen anderer Nutzer als Vorlage dienen und nicht jeder Nutzer muss sich eigene Filterregeln ausdenken,

Statistiken Um schnell zu entscheiden, ob ein Dump für einen bestimmten Anwendungsfall geeignet ist, sollten Statistiken über den Inhalt (z.B. die Anzahl der Entitäten) des Dumps angezeigt werden. Zusätzlich sollten auch schon während der Generierung Statistiken zum Fortschritt des Dumps bereitgestellt werden.

Nachvollziehbarkeit Gerade für den wissenschaftlichen Einsatz ist es erforderlich, dass die Herkunft der Daten und Generierung nachvollziehbar ist. Es sollte demnach leicht sichtbar sein, wie der Dump entstanden ist und eine Reproduktion des Dumps mit diesen Daten möglich sein.

3.2 nicht-funktionale Anforderungen

Neben den Anforderungen an die Funktion des Systems existieren auch eine Reihe von weiteren Anforderungen:

Hardwareanforderungen Der Ressourcenverbrauch des Systems sollte in einem akzeptablen Rahmen liegen. Als Anhaltspunkte für die Beurteilung dienen hier vergleichbare Systeme, wie bspw. der Wikidata Query Service, welcher ebenfalls Services basierend auf den RDF-Daten von Wikidata anbietet und gleichzeitig deutlich populärer ist. Demnach sollte das System entsprechend seiner Relevanz geringere Hardwareanforderungen als der Wikidata Query Service haben.

Freie Lizenz Die Wikimedia Foundation legt großen Wert darauf, möglichst viele der verwendeten Tools unter einer freien Lizenz bereitzustellen[@Pro13]. Wenn das System in der Wikimedia Cloud betrieben werden soll, dann ist die Veröffentlichung des Quellcodes unter einer freien Lizenz sogar Pflicht[@Pro19a].

Bearbeitungszeit Das System sollte maximal einen Tag zur Generierung eines Dumps benötigen, besser unter 12 Stunden. Während längerer Prozesse sollte keine ständig Verfügbarkeit des Nutzers erwartet werden.

Erweiterbarkeit Da die funktionalen Anforderungen an das System sehr allgemein sind, muss auf Erweiterbarkeit geachtet werden sodass neue Anforderungen einfach umgesetzt werden können. Es sind beispielsweise viele verschiedene Filtermöglichkeiten und Statistiken denkbar, die nicht alle in der ersten Version umgesetzt werden können. Es ist daher sinnvoll vor allem in diesem Bereich auf Erweiterbarkeit zu achten.

Skalierbarkeit Wikidata wächst beständig, aktuell existieren etwas weniger als 59 Millionen Items.cite Insgesamt gibt es über 700 Millionen Statements und diese Zahl ist allein im letzten Jahr um 200 Millionen gewachsen. Das System muss also mit dieser Menge an Daten umgehen können und dies auch für die nächsten Jahre noch leisten können.

3.3 Verwandte Arbeiten

SPARQL query service Linked Data Fragments Dbpedia Topical Dumps Wikidata REST API Wikidata Toolkit petscan/catscan

4 Design

4.1 Architektur

client only, dump processor (RDF vs JSON dumps), indexing, sparql based

4.2 Auswahl externer Dienste

Archivierung: Vergleich Anbieter (Zenodo) Hosting: auf toolforge

4.3 Interface

5 Implementierung

Die Anwendung besteht aus zwei Teilen: Backend und Frontend. Über das Frontend können Nutzer Aufträge für Dumps erstellen und existierende Dumps verwalten, während das Backend nur für die Generierung von Dumps zuständig ist. Das Backend stellt dazu in regelmäßigen Abständen Anfragen an eine Datenbank, um nach neuen Aufträgen zu schauen welche vom Frontend hinzugefügt wurden. Man könnte dafür auch eine spezielle Message-Queue verwenden. Die Datenbank wäre in diesem Fall allerdings trotzdem notwendig, um Metadaten zu den Aufträgen persistent zu speichern. Um die Komplexität gering zu halten wurde deshalb auf eine separate Message-Queue verzichtet.

In diesen Kapitel werden wir uns zunächst das Datenmodell anschauen, welches zur Kommunikation verwendet wird. Danach werden wir die Implementierung von jeweils Frontend und Backend genauer betrachten.

5.1 Datenmodell

Eine Übersicht des Datenmodells liefert Abb. 5.1. Das zentrale Element ist der Dump, welcher alle Metadaten zu einem Auftrag speichert. Neben ein paar einfachen Daten wie Titel (`title`) und Zeitpunkt der Erstellung des Auftrags (`created_at`) bzw. Fertigstellung (`finished_at`) ist jeder Dump durch eine JSON-Spezifikation (`spec`) charakterisiert. Zusätzlich hat der Dump auch Felder für Statistiken, wie die Anzahl der Entitäten (`entity_count`) und Dateigröße (`compressed_size`).

Für jeden Durchlauf des Backends wird ein `Run` angelegt. Jeder Durchlauf verarbeitet dazu mehrere Aufträge. Damit der aktuelle Fortschritt ermittelt werden kann, wird die Anzahl der bereits verarbeiteten Entitäten in dem Attribut `count` gespeichert. Da die Anzahl von Entitäten in dem vollem Wikidata-Dump bekannt ist, lässt sich daraus der Fortschritt errechnen.

[TODO describe spec]

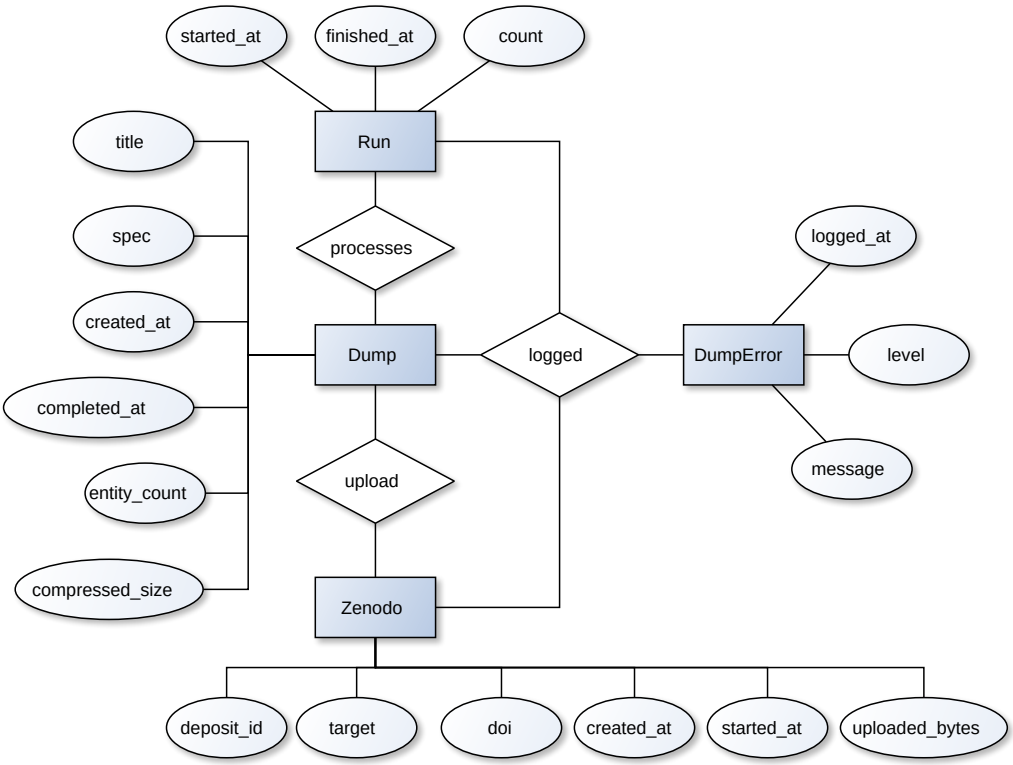

Abbildung 5.1: Datenbankschema

5.2 Backend

Für die Implementierung des Backends wird das Wikidata-Toolkit verwendet. Da Wikidata-Toolkit in Java geschrieben ist, muss deswegen auch eine Java-kompatible Programmiersprache verwendet werden. An dieser Stelle wurde Java gewählt, um auch die Wartbarkeit der Anwendung in Zukunft sicherzustellen, da Java im Vergleich zu anderen Programmiersprachen mit Java-Kompatibilität (wie zum Beispiel Scala[1] oder Kotlin[2]) deutlich weiter verbreitet und bekannter ist.

Die Hauptschleife des Backends besteht aus zwei Phasen, Warten und Verarbeitung. Während der wartenden Phase wird kontinuierlich auf neue Aufträge gewartet. Sobald neue Aufträge verfügbar sind, wird die Verarbeitung gestartet. Neue Aufträge werden dann erst wieder nach Beendigung des aktuellen Verarbeitungsprozesses abgerufen.

Am Start befindet sich das Backend in der wartenden Phase. Um neue Aufträge abzurufen, werden dazu periodische die in Listing 5.1 dargestellten SQL-Befehl in einer Transaktion ausgeführt. Da in Zeile 6 nur Aufträge zugewiesen werden, die noch keinem `Run` zugwiesen sind, kann diese Abfrage theoretisch auch von mehreren Prozessen gleichzeitg ausgeführt werden ohne dabei Kollisionen zu erzeugen. Es ist so unmöglich, dass ein Auftrag mehr als einem `Run` zugwiesen wird, was die Robustheit des Systems erhöht. Falls diese Befehle keine Ergebnisse liefern (wenn keine neuen Aufträge vorliegen), wird eine definierte Zeit gewartet bevor dieser Vorgang wiederholt wird. Diese Wartezeit führt gleichzeitig dazu, dass mehrere Aufträge gesammelt werden können, da die Verabeitung nicht direkt beginnt.

```
1 -- neuen Run erstellen
2 INSERT INTO run () VALUES ()
3 -- generated id: 1
4
5 -- Aufträge dem Run zuweisen
6 UPDATE dump SET run_id = 1 WHERE run_id IS NULL
7
8 -- Zugewiesene Aufträge abrufen
9 SELECT id, spec FROM dump WHERE run_id = :run
```

Listing 5.1: Abrufen neuer Aufträge

Zum Verarbeiten der Aufträge wurde der in Wikidata-Toolkit bereits vorhandene RDF-Export angepasst. Der RDF-Export ist dabei als eine Klasse implementiert, wel-

[1] https://www.scala-lang.org/
[2] `https://kotlinlang.org/`

che das von Wikidata-Toolkit erwartete Interface `EntityDocumentProcessor` implementiert (Listing 5.2).

```
public interface EntityDocumentProcessor {
  void processItemDocument(ItemDocument itemDocument);
  void processPropertyDocument(PropertyDocument propertyDocument);
  void processLexemeDocument(LexemeDocument lexemeDocument);
}
```

Listing 5.2: EntityDocumentProcessor Interface

Listing 5.3 zeigt den Ablauf des Exports in Pseudocode. Für jede Entität (Items, Properties und Lexemes) wird dazu zunächst überprüft ob sie überhaupt exportiert werden soll. Nur wenn das der Fall ist, werden danach für jedes Statement die Optionen zum Export entsprechend der Filter-Spezifikation bestimmt. Wenn die Optionen feststehen, kann dann das Statement exportiert werden. Aktuell werden Lexemes noch nicht untersützt, da Wikidata-Toolkit den RDF-Export dafür noch nicht implementiert hat. Wenn ein Lexeme exportiert werden soll wird deshalb ein Fehler erzeugt. Zusätzlich zu den Statements wird noch RDF für Labels, Descriptions, Aliases, Sitelinks und Metadaten zu der Entität erzeugt, falls entsprechend der Filter-Spezifikation verlangt.

```
for each entity:
  if spec includes entity:
    if entity is lexeme: raise error

    if spec.labels: export entity labels
    if spec.aliases: export entity aliases
    if spec.descriptions: export entity descriptions

    for each statement:
      let options = get options for statement from spec
      export statement with options

    if spec.sitelinks: export entity sitelinks
    if spec.meta: export entity metadata
```

Listing 5.3: Export Pseudocode

Wikidata-Toolkit unterstützt mehrere `EntityDocumentProcessors` gleichzeitig. Damit können mehrere Aufträge in einem Durchlauf verarbeitet werden. Diese Funktionalität wird auch verwendet, um den aktuellen Fortschritt des Durchlaufs in der Datenbank zu aktualisieren. Dazu zählt ein `EntityDocumentProcessor` die Anzahl der verarbeiteten Entities mit und speichert diese regelmäßig in der `run`-Tabelle der Datenbank.

[TODO Auf StatementOptions / Spec interface eingehen]

5.3 Frontend

Das Frontend besteht aus einem Web-Interface zum Erstellen von Dump-Aufträgen und Verwaltung der existierenden Dumps. Für dessen Implementierung wird hauptsächlich HTML/CSS mit Typescript verwendet, für die Auslieferung und Kommunikation mit der Datenbank gibt es aber auch eine kleine serverseitige Anwendung, welche in Python mit Flask geschrieben ist.

Der serverseitige Teil bietet dazu Endpunkte für das Erstellen, Suchen, Herunterladen und Abfrage von Informationen von Dumps an. Dazu werden vier Endpunkte bereitgestellt:

- `POST /create` erstellt einen neuen Dump-Auftrag. Der Request-Body werden die Filter-Spezifikation sowie ein paar Metadaten (Titel, etc.) übergeben.
- `GET /dump/<id>` liefert eine Statusseite mit Informationen zu einem bestimmten Dump.
- `GET /dumps` gibt eine Liste aller Dumps zurück.
- `GET /download/<id>` lädt einen Dump herunter.

6 Evaluation

7 Fazit

Literatur

[VK14] Denny Vrandečić und Markus Krötzsch. “Wikidata: A Free Collaborative Knowledge Base”. In: *Communications of the ACM* 57 (2014), S. 78–85 (siehe S. 2).

Webpages

[@Pro19a] Wikimedia Project. *Wikitech:Cloud Services Terms of use*. 2019. URL: https://web.archive.org/web/20190804123117/https://wikitech.wikimedia.org/wiki/Wikitech:Cloud_Services_Terms_of_use (besucht am 4. Aug. 2019) (siehe S. 4).

[@Pro13] Wikimedia Foundation Project. *Resolution: Wikimedia Foundation Guiding Principles*. 2013. URL: https://web.archive.org/web/20190804122555/https://foundation.wikimedia.org/wiki/Resolution:Wikimedia_Foundation_Guiding_Principles (besucht am 4. Aug. 2019) (siehe S. 4).

[@Pro19b] Wikimedia Foundation Project. *Wikibase/Indexing/RDF Dump Format*. 2019. URL: https://web.archive.org/web/20190801131618/https://www.mediawiki.org/wiki/Wikibase/Indexing/RDF_Dump_Format (besucht am 1. Aug. 2019) (siehe S. 2).