Richtlinie zum „Software-Sprint“

OSM-Generalisierung – Generalisierung von OpenStreetMap-Daten mit osm2pgsql

Schlussbericht

Zuwendungsempfänger:

Jochen Topf

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01IS22S29 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Kurze Darstellung der Aufgabenstellung und Motivation

Karten sind ein wichtiges Werkzeug, um die Welt um uns herum zu verstehen. Dazu reicht es aber nicht, auf die direkte Umgebung zu schauen. Um große Räume und Zusammenhänge zu verstehen braucht es Übersicht. Dazu müssen aus den vielen Details in den Rohdaten diejenigen Informationen extrahiert werden, die für Übersichtskarten nötig sind. Bei klassischen Karten braucht das viel Handarbeit der Kartographen. In der modernen Welt mit Kartendaten, die sich von Minute zu Minute ändern können und die wir auf ganz verschiedene Arten darstellen wollen, muss das automatisiert passieren.

OpenStreetMap als größte freie weltweite Geodatenbasis lebt davon, dass man schöne und nützliche Karten daraus erzeugen kann. Für die Motivation der Aktiven ist es entscheidend, dass sie "ihre" Daten auch schnell auf der Karte sehen können. Daher ist es besonders wichtig, Tools zu schaffen, die die Daten auch entsprechend schnell und mit begrenztem Rechenaufwand in Karten umsetzen können.

Das Open-Source-Tool *osm2pgsql* ist ein wichtiger Teil vieler Toolchains, die OpenStreetMap-Daten auf Karten bringt. Es wird für viele Karten verwendet, unter anderem für die bekannteste OpenStreetMap-Karte auf der Webseite openstreetmap.org. Auf größeren Maßstäben kann man die Daten von *osm2pgsql* direkt verwenden. Aber für kleinere Maßstäbe müssen die Daten erst sinnvoll ausgewählt und zusammengefasst werden. Das hilft einerseits dabei, die Datenmenge zu reduzieren, um die Karten schnell genug zeichnen zu können, und andererseits dabei kartographisch ansprechende Ergebnisse zu erzielen. Diesen Vorgang der Auswahl und Zusammenfassung nennt man (kartographische) Generalisierung. Generalisierung ist ein komplexes Thema und, je nach Art der Daten und der erwünschten Ergebnisse, gibt es viele verschiedene Ansätze, wie man so eine Generalisierung durchführen kann. Im Projekt sollte *osm2pgsql* so erweitert werden, dass es einige dieser Generalisierungsschritte durchführen kann.

Das Projekt bestand aus zwei Teilen die ineinandergreifen: Zunächst sollten verschiedene Algorithmen (zum Beispiel zur Polygonvereinfachung oder zum bestimmen und priorisieren von

Beschriftungspunkten) prototypisch in SQL und/oder in C++ umgesetzt werden, um ihre Performance zu analysieren. Dabei konnte teilweise auf vorhandene Software zurückgegriffen werden, insbesondere bietet die PostGIS-Erweiterung von PostgreSQL viele nützliche Bausteine an.

Im zweiten Teil sollte dann *osm2pgsql* entsprechend erweitert werden, dass es diese Generalisierungsschritte beim Import der Daten anwenden kann. Außerdem sollen bei Änderungen der Ursprungsdaten nur die abgeleiteten Daten aktualisiert werden, für die das nötig ist. Diese Verarbeitung soll auf verschiedene Arten geschehen:

1. Erweiterung des existierenden Expire-Mechanismus in *osm2pgsql* um die kachelweise Generalisierung von geänderten Daten.

2. Speichern von eindeutigen IDs an den generalisierten Daten und Update aller generalisierten Daten, deren Quell-Objekte sich geändert haben.

3. Aktualisierung aller Daten eines bestimmten Layers,wenn das schnell genug geht und eine Verarbeitung wie in (1) oder (2) zu aufwändig wäre.

Als Meilensteine waren geplant:

1. Prototypische Entwicklung einiger Generalisierungsalgorithmen.

2. Evaluation der Performance und Auswahl geeigneter Ansätze für die weitere Entwicklung.[1]

3. Erweiterung von osm2pgsql um die nötigen Mechanismen für das erweiterte Änderungsmanagement.

4. Einbindung der in 2. ausgewählten Algorithmen in *osm2pgsql*.

5. Dokumentation der neuen Möglichkeiten und Release einer neuen Softwareversion.

Beitrag des Projektes zu den Zielen der Förderinitative „Software-Sprint“

Karten formen unseren Blick auf die Welt. Eine Karte kann die Wirklichkeit immer nur teilweise wiedergeben. Wer eine Karte erstellt, entscheidet immer auch, was darauf erscheinen soll und was weggelassen wird und wie genau die Darstellung aussieht. So wie Geobasisdaten offen sein sollen, damit jede Bürgerin und jeder Bürger mit-verstehen, mit-reden und mit-handeln kann, so muss auch die Software, die zur Herstellung der Karten verwendet wird, in der Hand der Nutzer sein.

Wir dürfen die Verarbeitung der Daten nicht nur proprietärer Software und geheimen Algorithmen überlassen, die wir nicht verstehen und auf die wir keinen Einfluss haben. Neben offenen Geodaten brauchen wir Open-Source-Software, die die Daten in Karten umsetzt und auch Algorithmen zur Generalisierung enthält, die jede und jeder nutzen kann. Das soll es allen, vom Studenten zur Verwaltungsmitarbeiterin zum Aktivisten in der Zivilgesellschaft und zur Wirtschaft ermöglichen, die Karten zu bauen, die sie brauchen.

1 https://blog.jochentopf.com/2022-11-07-selection-of-generalization-problems.html

Ausführliche Darstellung der Ergebnisse

Im Projekt ist ein **Framework** entstanden, das steuert, wie Generalisierungsalgorithmen verschiedener Art auf die Daten angewendet werden können[2]. Anders als ursprünglich geplant ist der Code nicht direkt in das Tool *osm2pgsql* eingebunden worden, sondern ein zweites Tool *osm2pgsql-gen* entstanden, dass aber zum gleichen Projekt[3] gehört und auch die selbe Konfigurationsdatei liest. Da der Code aber noch recht neu ist und wenig Erfahrung damit vorliegt, wurde in Rücksprache mit den anderen Entwicklern von *osm2pgsql* beschlossen, den Code vorerst etwas getrennt zu halten. Die Dokumentation wurde aber schon integriert[4]. In der Funktionalität macht das keinen wesentlichen Unterschied, allenfalls ist die Benutzung etwas schwieriger, weil man das mit den beiden Programmen im Auge behalten muss.

Bis zum Projektabschluss sind einige Algorithmen komplett und einige teilweise umgesetzt worden:

Für die Ableitung von **Beschriftungspunkten für Polygone** (oder allgemeiner: „representativen Punkten für Polgyone“) wurde ein Algorithmus zur Berechnung des *Pole of Inaccessibility* umgesetzt[5]. Anders als die anderen Algorithmen wird dieser Code bereits beim Import ausgeführt und ist bereits in der aktuellen Release (1.8.1) von *osm2pgsql* enthalten.

Für die **Zusammenfassung und Vereinfachung von Polygonen** wurden zwei Versionen des letztlich gleichen Algorithmus implementiert. Die eine Version arbeitet direkt auf Vektor-Daten, die andere konvertiert die Daten erst in ein Raster und macht dann die Vereinfachung. Beide kommen zu ähnlichen Ergebnissen, die Raster-Verarbeitung ist aber deutlich schneller.[6] Diese Algorithmen werden üblicherweise für Polygone mit Landnutzungsinformationen (Wald, Felder, bebautes Gebiet, Wasserflächen, …) eingesetzt. Die Algorithmen verarbeiten die Daten kachelweise (als „tiles“), auch die Updates sind so einfach und schnell möglich. Die Implementierung ist bereits recht stabil.

Ein Sonderfall der Vereinfachung von Polygonen ist die **Ableitung von Siedlungsgebieten** aus Landnutzungs-Polygonen einerseits, aber auch Gebäudeumrissen und Straßennetzen. Die Landnutzungspolygone sind in OSM nicht überall erfasst, sodass diese weiteren Daten zur Hilfe genommen werden. Die Implementierung funktioniert in meinen Tests recht gut.

Zur **Darstellung von Flußnetzwerken** muss ermittelt werden, welche Flüsse „wichtig“ sind und nur diese werden dann geometrisch vereinfacht und dargestellt[7]. Dazu ermittelt der Algorithmus die Breite der Flüsse als Metrik für ihre Bedeutung. Das funktioniert allerdings noch nicht ganz zufriedenstellend, wenn Flüsse sich in viele Äste aufteilen. Weitere Arbeiten werden zeigen, ob hier ein anderer Ansatz gewählt werden muss.

---

2 https://blog.jochentopf.com/2022-11-15-processing-architecture-for-generalization-in-osm2pgsql.html

3 https://github.com/openstreetmap/osm2pgsql

4 https://osm2pgsql.org/doc/manual.html#generalization

5 https://blog.jochentopf.com/2022-11-10-finding-representative-points-for-polygons.html

6 https://blog.jochentopf.com/2022-11-21-generalizing-polygons.html

7 https://blog.jochentopf.com/2023-01-30-generalizing-river-networks.html



Figure 1: Vorher: Unzureichende Auswahl von Städten (Quelle: osm.org)



Figure 2: Bessere Auswahl von Städten mit dem Discrete Isolation Algorithmus.

Zur **Auswahl von Städten nach Bedeutung** wurde der „discrete isolation"-Algorithmus implementiert.[8] Damit wird statt der absoluten Bedeutung einer Stadt (gemessen zum Beispiel anhand der Bevölkerungszahl) eine relative Bedeutung ermittelt, damit die Karte nicht in einigen Bereichen komplett leer ist, während sich in dichter besiedelten Gegenden die Städtenamen den Platz streitig machen. Der gleiche Algorithmus kann auch für andere Fälle eingesetzt werden, zum Beispiel um nur die Berggipfel zu markieren, die stark aus ihrer Umgebung herausragen. Die Implementierung ist schnell und stabil und die Daten können für die ganze Welt in wenigen Minuten berechnet werden.

Außer dem etwas experimentelleren Algorithmus für die Vereinfachung von Flußnetzwerken sind damit alle geplanten Algorithmen komplett umgesetzt worden und in der Anwendung verfügbar.



Figure 3: Beispielergebnis aus dem Projekt: Europakarte mit generalisierten Landcover- und Siedlungsflächen und Auswahl von Städten. (Die Wasserflächen sind nicht vereinfacht.)

Das Projekt hat gezeigt, dass die Generalisierung mit vertretbarem Aufwand möglich ist. Typischerweise braucht ein Import aller OSM-Daten für die ganze Erde mit *osm2pgsql* etwa 10

8 https://blog.jochentopf.com/2022-12-19-selecting-settlements-to-display.html

Stunden. (Das hängt natürlich aber von der Hardware und der Konfiguration ab.) Die Generalisierungsschritte, die im Projekt umgesetzt wurden, brauchen dann weitere ca. 5 Stunden. Für eine komplettes Kartenbild sind sicher noch weitere Schritte nötig, aber es sind sicher auch noch Optimierungen im Code möglich. Eine Gesamtlaufzeit von unter einem Tag erscheint damit realistisch, womit wir deutlich besser dastehen, als bisher genutzte Verfahren, bei denen die Verarbeitung mehrere Tage braucht und die dabei noch weniger können.

Um bei Änderungen der Daten die Kacheln zu ermitteln, die neu berechnet werden müssen, wurde im Rahmen des Projektes der bestehende Code in diesem Bereich erweitert und flexibler gemacht. Bisher war es nur möglich einen Satz von zu ändernden Kacheln zu definieren, für diese Anwendung ist es aber nötig, für jeden Layer und gegebenenfalls Zoom-Level einen eigenen Kachel-Satz zu definieren. Dieser Code ist auch bereits in den Hauptteil von *osm2pgsql* eingeflossen, weil er auch in anderem Kontext nützlich ist.[9] In unserem Kontext ist es damit möglich, Änderungen an den Rohdaten effizient in Änderungen der generalisierten Daten einfließen zu lassen.

Zielgruppe, Nutzen und mögliche Weiterentwicklungen

Die Ergebnisse des Projektes sind, zumindest für *early adopters*, jetzt direkt nutzbar, das heißt sie können die Generalisierungsfunktion für ihre Karten nutzen. Wir brauchen aber noch mehr Erfahrungen von den Nutzern in verschiedenen Umgebungen und für verschiedene Karten, um zu sehen, wie gut und stabil das alles in der Praxis läuft. In erster Linie wollen wir die neuen Funktionen jetzt allgemeiner bekannt machen, sodass Kartographinnen und Kartographen, die neuen Funktionen in ihren Konfigurationen austesten.

Die nun vorhandenen spezifischen Generalisierungsalgorithmen sind aber nur ein Anfang. Das Framework ist vorhanden, sodass auch andere Entwicklerinnen und Entwickler jetzt einfacher weitere Algorithmen umsetzen und ergänzen können. Es ist zu hoffen, dass die neuen Möglichkeiten jetzt auch mehr Kartographen dazu bringen, die Möglichkeiten der Generalisierung für ihre Karten in Betracht zu ziehen. Das gilt einerseits für Anwenderinnen und Anwender, die schon *osm2pgsql* nutzen, aber vielleicht gibt es ja auch den Anstoß für das eine oder andere Projekt von ihren bisher genutzten Tools auf *osm2pgsql* zu wechseln oder Generalisierung auch in andere Projekte einzubauen. Wir werden sicher im Rahmen der Softwarepflege im *osm2pgsql*-Projekt an Bugfixes und neuen Algorithmen arbeiten.

Kurze Darstellung der Arbeiten, die zu keiner Lösung geführt haben

Im Laufe der Arbeiten am Meilenstein 1 und 2 hat sich herausgestellt, dass sehr viele der Generalisierungsalgorithmen entweder schnell genug auf den kompletten Daten laufen können, wenn man sie in C++ implementiert, oder dass der kachelbasierte Ansatz („tiles“) zur Aufteilung der Daten gut funktioniert. Ich habe daher auf den ursprünglich geplanten Ansatz mit Verfolgen geänderter Objekte über ihre Id verzichtet, was mir mehr Zeit für die anderen Ansätze gebracht hat. Eine Erweiterung in diese Richtung kann in Zukunft ergänzt werden, wenn sich das doch als nötig erweisen sollte.

9 https://blog.jochentopf.com/2023-02-25-tile-expiry-and-generalization.html

Kurze Angabe von Präsentationsmöglichkeiten für mögliche Nutzer

Speziell für dieses Projekt und die weitere Arbeit dazu wurde auf unserer Projektwebseite osm2pgsql.org der Bereich https://osm2pgsql.org/generalization/ eingerichtet. Dort gibt es Links zu diversen Blog-Artikeln, die die verschiedenen Algorithmen und das Vorgehen im Projekt beschreiben. Aller Code wurde bereits im *master*-Branch auf Github unter https://github.com/openstreetmap/osm2pgsql eingebunden und wird auch im nächsten Release zur Verfügung stehen. Interessenten können auch die ausführliche Dokumentation unter https://osm2pgsql.org/doc/manual.html#generalization lesen. Auf der FOSSGIS-Konferenz im März 2023 werde ich einen Vortrag über das Projekt halten, der danach Online zur Verfügung steht.

Kurze Erläuterung zur Einhaltung der Arbeits- und Kostenplanung

Der Plan konnte im Großen und Ganzen eingehalten werden, es gab nur leichte Verschiebungen wegen Urlaubszeiten.

Kurze Darstellung von etwaigen Ergebnissen bei anderen Stellen

Gespräche mit anderen OpenStreetMap-Aktiven und Interessenten an der Generalisierung haben mich bei der Auswahl interessanter Fragestellungen und der Auswahl der dann umgesetzten Algorithmen beeinflußt.