Untersuchung von Data Science Projekten auf GitHub

Jan Patrick Holler

Hochschule Darmstadt - University of Applied Science

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbmn
FACHBEREICH MATHEMATIK
UND NATURWISSENSCHAFTEN

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Problemstellung

Im Zeitalter der großen Datenmengen wird der Bereich Data Science immer relevanter. Die Aufgabenbereiche eines Data Scientists sind weitreichend und es existieren viele verschiedene Stellenbeschreibungen. Doch wie erfolgt tatsächlich die Aufgabenverteilung in Projekten? Werden spezifische Rollen eingenommen oder erfolgt keine Trennung der Aufgabenbereiche?
Im Rahmen dieser Masterarbeit werden öffentlich zugängliche Data Science Projekte von der Plattform GitHub analysiert. Hierfür werden spezifisch Projekte der IPython Umgebung Jupyter betrachtet.
Ziel ist es herauszufinden, ob in gemeinsam bearbeiteten Projekten eine Rollenverteilung erkennbar ist. Hierfür wird der geschriebene Source Code als Kriterium verwendet.

Abstract Syntax Tree

Zur Analyse des Source Codes können, wie auch bei Texten, Methoden des Natural Language Processing angewendet werden. Hierbei wird der Source Code als Text betrachtet. Um den syntaktischen Zusammenhang des Codes zu betrachten, wird dieser in einen Syntax Baum umgewandelt. Dies geschieht mit Hilfe des in Python enthaltenen AST Moduls.
Der Abstrakte Syntaxbaum zeichnet sich durch seine kompakte Struktur und Plattformunabhängigkeit aus. Nur die wesentlichen Inhalte eines Ausdruckes werden abgebildet, konkreter Syntax wird hierbei weggelassen.
Abbildung 1 zeigt wie ein solcher AST für einen Python Ausdruck aussehen kann. Es gibt verschiedene Knotentypen unter Anderem für Imports, Zuweisungen und Funktionsaufrufe. Zur weiteren Analyse werden speziell die Funktionsaufrufe betrachtet und als Repräsentation des geschriebenen Source Codes verwendet.

Module Matching

In den AST Knoten für Funktionsaufrufe sind zwar Informationen darüber enthalten, wie die Funktion heißt und auf welcher Variable diese ausgeführt wird, jedoch nicht aus welchem Modul diese Funktion stammt. Um dieses Problem zu lösen, werden Funktion manuell den zugehörigen Modulen zugeordnet.
Hierfür wurde eine Klasse entworfen, die jeden einzelnen Knoten betrachtet und Informationen über importierte Module, aufgerufene Funktionen und Variablen Zuweisungen sammelt.

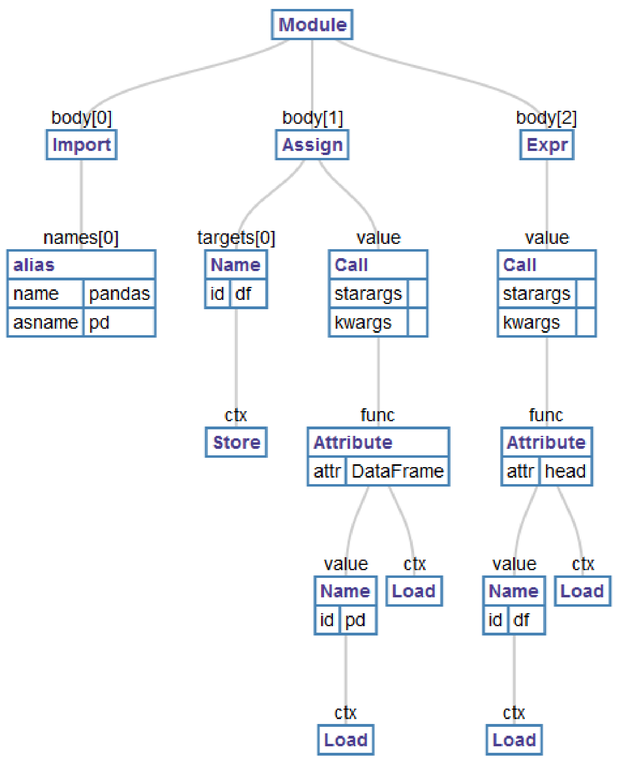

Figure 1: AST Beispiel

Es sind folgende Anforderungen ans Module Matching gestellt:

1. Es müssen verschiedene Formen des Imports korrekt erkannt werden.
2. Es müssen verschiedene Formen des Funktionsaufrufes korrekt erkannt werden.
3. Eine Verknüpfung zwischen importierten Modulen und aufgerufenen Funktionen muss hergestellt werden.
4. Auch Funktionsaufrufe die nicht direkt von dem Import-Objekt aufgerufen werden müssen zugeordnet werden.
5. Es soll Python Source Code verarbeiten und eine Liste mit den Zuordnungen zurückgeben.

Nachdem alle Knoten abgearbeitet wurden erfolgt das Matching. Hierfür werden Namen der Variablen mit den Namen der importierten Module abgeglichen und über Zuweisungen mit den Funktionen verknüpft.

Daten

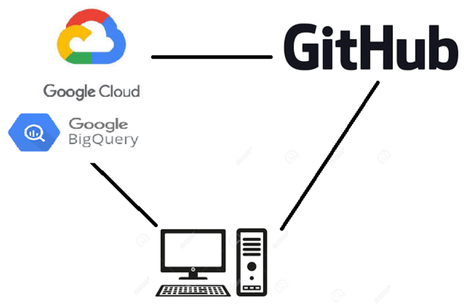

Figure 2: Datenarchitektur

Als Datengrundlage für diese Arbeit werden Projekte der Plattform GitHub betrachtet. Diese stellt einen Datensatz über alle öffentlichen Projekte auf der Plattform Google BigQuery zur Verfügung. Unter Verwendung von SQL Queries werden die wesentlichen Projekte gefiltert. Anhand dieser gesammelten Informationen werden die relevanten Repositories direkt von GitHub lokal abgespeichert. Anhand der Commit Änderungen wird der geschriebene Source Code extrahiert. Hieraus werden wiederum die aufgerufenen Funktionen extrahiert und den Autoren zugewiesen. Weiterhin werden die Funktionen durch das Module Matching den entsprechenden Modulen zugewiesen.
Hieraus ergibt sich der Datensatz, der für die weitere Analyse verwendet wird.

Topic Modeling

Topic Modeling ist ein Analyseverfahren, dass Dokumente analysiert und anhand der verwendeten Wörter in Themen unterteilt. Für das Topic Modeling wurden die einzelnen Autoren als Dokumente definiert. Die verwendeten Funktionen sind hierbei die Wörter im Dokument. Somit erfolgt eine Unterteilung der Autoren anhand der verwendeten Funktionen.
Zur Durchführung des Topic Modeling muss zunächst ein Wörterbuch angelegt werden, der alle Funktionsnamen enthält. Anhand des Wörterbuchs werden die Dokumente durch Verwendung eines Algorithmus verglichen. Hierfür wird als Algorithmus Latent Dirichlet Allocation (LDA) verwendet. Die Unterteilung erfolgt anhand der Worthäufigkeiten, wobei einem Dokument eine Wahrscheinlichkeitsverteilung über mehrere Themen zugewiesen wird.
Ein Thema wird anhand einer Wortverteilung repräsentiert. Abbildung 3 zeigt eine Visualisierung eines Themas des finalen Topic Modeling. Hierbei wird die Wortverteilung eines bestimmten Themas dargestellt. Es ist eine klare Unterteilung anhand der Funktionen sichtbar. Innerhalb der Themen sind viele Funktionen enthalten, die dem selben Modul zugeordnet sind.

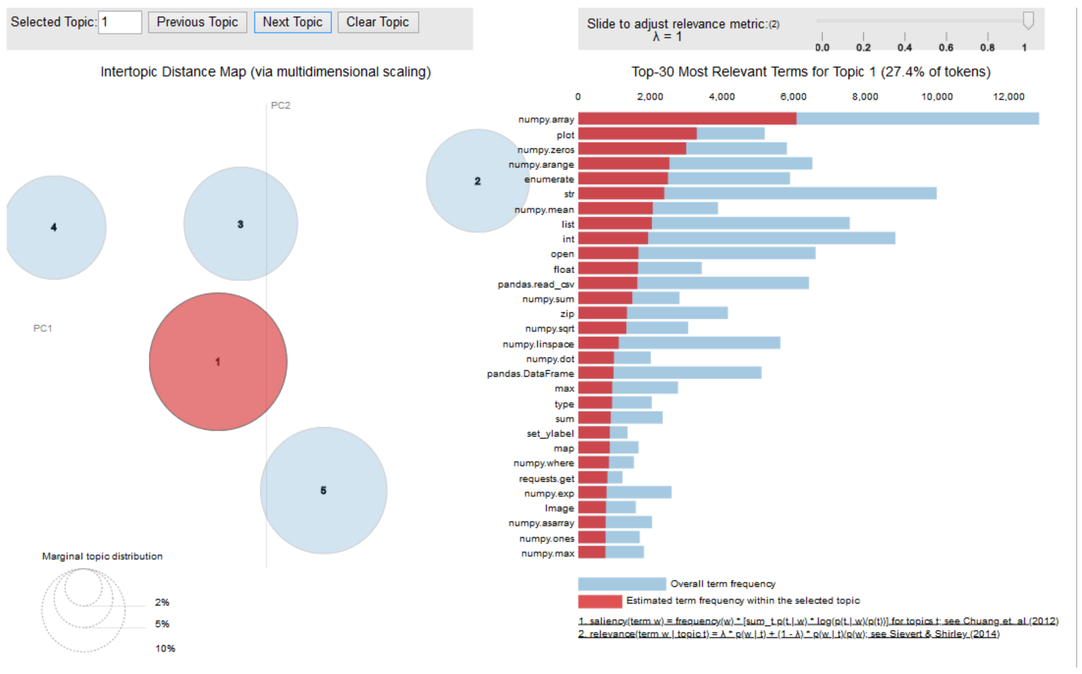

Figure 3: Topic Modeling

Ergebnis

Im Rahmen dieser Arbeit konnte zwar eine Differenz zwischen aufgerufenen Funktionen nachgewiesen werden, klare Rollen lassen sich jedoch nicht ableiten. Häufig verwendete Module verfälschen das Ergebnis, da sie mehrere Themen repräsentieren. Im Ausblick auf weitere Forschung kann untersucht werden, wie die Verteilungen sich ändern, wenn solche Module herausgefiltert werden.
Weitergehend wurde festgestellt, dass das Matching zwischen Modul und Funktion nicht immer reibungslos gelingt. Dies liegt der verschiedenen Arten der Funktionsaufrufe zu Grunde. Hierfür muss die Abfolge von Knoten betrachtet werden um die Fälle abdecken zu können.
Die spezielle Struktur der Jupyter Notebooks erschwert die maschinelle Verarbeitung. Der Source Code ist nicht in reiner Form enthalten, sondern in einer JSON Struktur eingenistet. Durch die notwendige Umwandlung in reinen Python Code gehen einige Daten aufgrund von Versionsunterschieden und fehlerhafter Struktur verloren.