Communities im Co-Author-Graphen

Tobias Schmid, Andreas Knöpfle

Fachbereich Informatik
Freie Universität Berlin
Anschrift
Postleitzahl Ort
emaiaddresse@autor1
emaiaddresse@autor2

Abstract: Es folgt ein kurzer Überblick über die Arbeit.

1 Einleitung

Der DBLP Datensatz beinhaltet eine große Sammlung wissenschaftlicher Publikationen und deren Meta-Daten aus dem Bereich Informatik. Insgesamt sind bereits mehr als 1,8 Millionen Publikationen aufgenommen worden ([**?**]). Aus diesen Daten lässt sich ein Co-Autor-Graph ableiten, welcher alle Autoren als Knoten und Publikationen zwischen ihnen als gemeinsame Kanten beschreibt. Dieser Graph kann durch graphentheoretische Betrachtungen analysiert werden.

Anschaulich heißt das: Haben zwei Autoren miteinander publiziert, sind deren Knoten durch eine Kante verbunden. Das Gewicht einer Kante zeigt zusätzlich die Anzahl der gemeinsamen Publikationen an.

Bei einem Datensatz dieser Größe könnte eine Analyse dieses Graphen repräsentative Aussagen über die Autoren und deren Umfeld geben. Somit wird die wissenschaftliche Fragestellung dieser Arbeit wie folgt festgelegt:

Welche Charaktereigenschaften weisen die Autoren und ihr Umfeld im Co-Autorgraph der DBLP auf?

Um dieser Frage nachzukommen, wird eine Infrastruktur entwickelt, die es ermöglicht den Co-Autor-Graphen zu generieren und zu visualisieren. Zusätzlich müssen Werkzeuge entwickelt werden, die es erlauben den Graphen zu analysieren.

2 Ausgangssituation

Als Datenquelle für den Co-Autor-Graphen werden die Metadaten der DBLP Datenbank im XML-Format benutzt [**?**]. Die Metadaten bestehen aus einer Liste der wissenschaftlichen Artikel mit Titel, Datum, Autoren und zusätzlichen Informationen über die Veröffentlichung.

In **??** ist ein Ausschnitt aus dieser XML Quelle abgebildet. Aus dieser Quelle kann der Co-Autor-Graph erzeugt werden, da jeder Artikel alle beteiligten Autoren enthält und somit auf Knoten und Kanten im Graphen abgebildet werden kann.

3 Randbedingungen

Um den Co-Autor-Graphen zu analysieren, wird eine effiziente Form des Datenzugriffs benötigt. Um diesen Zugriff technisch umzusetzen wird im Weiteren die Graphdatenbank Neo4J verwendet. Mit Hilfe von Neo4J kann der Graph einfach und effizient gespeichert und traversiert werden. Zusätzlich können Meta-Informationen zu den Knoten und Kanten in der Datenbank verwaltet werden, womit diese durch Analyseergebnisse angereichert werden können. Weiterhin unterstützt Neo4J die Entwicklung von Algorithmen auf den Graphen und bietet bereits einige Implementierungen dieser.

4 Communities im Co-Autor-Graphen

Die Autoren sind durch gemeinsame Publikationen im Co-Autor-Graphen miteinander verbunden. Viele Publikationen zwischen zwei Autoren lassen auf eine starke Verbindung zwischen diesen schließen. Durch diese Verbindungen können sich Gemeinschaften unter den Autoren bilden.

Im Folgenden wird also untersucht, ob im Co-Autor-Graphen der DBLP Gemeinschaften unter den Autoren zu finden sind.
Sollten Gemeinschaften gefunden werden können, ergeben sich die weiteren Fragestellungen:

Wie stark unterscheiden sich die Communities in ihrer Größe ?
Wie lassen sich die Communities noch anders charakterisieren (nicht nach der Größe) ?
Ist ein Zusammenhang zwischen Größe und Charakter einer Community erkennbar ?

Um diese Gemeinschaften, weiterhin mit dem englischen Begriff Communities bezeichnet, finden zu können, wird der in [**?**] beschriebene Community-Detection Algorithmus eingesetzt. Dieser Algorithmus findet Communities und deren Zusammensetzung als Hierarchiestruktur. Die Ergebnisse des Community-Detection Algorithmus werden in der Neo4J Datenbank gespeichert.

Die oberste ebene der Hierarchiestruktur, die die Zusammensetzung der Communities beschreibt, besteht aus den gefundenen Communities (Top-Level-Communities). Jede dieser Communities besteht wieder aus kleineren Communities und diese wieder aus Communities usw.. So gibt es mehrere Stufen, wobei die unterste Ebene die Autoren darstellt.

Die Größe jeder Top-Level-Community und ihrer Untergruppen kann durch einen rekursiven Algorithmus berechnet werden.

Die Conductance einer Community zeigt das Verhältnis der Anzahl der Kanten, welche diese Community mit einer anderen verbindet, zu der Anzahl der Kanten innerhalb der Community an. Dies kann als zusätzliche Charakterisierung verwendet werden.

4.1 Der Community Algorithmus

TODO ...

4.2 Die Messungen

Im Folgenden werden die Messungen näher beschrieben. Dabei wird zuerst auf die Messung der Community Größen und danach auf die Messung der Conductance eingegangen.

4.3 Größe

Da die Communities Hierarchisch aufgebaut sind, können die Größen der Communities auf den verschiedenen Hierarchie-Ebenen gemessen werden. Dazu sollte die Größe der unterste Ebene (eine ebene über der Autor Ebene) gleich der Anzahl der Autoren sein, welche direkt mit dieser Community verbunden sind. Die Größe jeder Community in einer höheren Ebene setzt sich dann durch die Anzahl der beinhaltenden Autoren der Unterebenen zusammen. So kann für alle Communities auf allen Ebenen die genaue Größe bestimmt werden.

Um diese Größen für den Co-Author-Graphen zu bestimmen, wird ein rekursiever Algorithmus entwickelt, welcher auf den einzelnen Ebenen für die Communities die Größen berechnet. In folgendem Pseudocode-Listing ist der rekursive Algorithmus abgebildet.

```
HashMap<Long, Object> counts;

long count(Node community)
-- Wenn Community Untercommunities hat
---- long counter_community
---- Für jede Untercommunity in Community
------ long count = count(Untercommunity)
------ counter_community += count
---- counts.put(community.getId(), counter_community)
---- return counter_community
--Sonst
---- return 1
```

Der Algorithmus startet bei dem ersten Community Knoten welcher der Methode übergeben wird. Es werden alle *BELONGS_TO* Beziehungen verwendet, um alle Unter-Communities des Community Knotens zu finden. dann wird der Algorithmus rekursiv auf alle Unter-Communities angewandt. Wenn der Algorithmus bei einem Autor Knoten angekommen ist, wird für diesen Autor Knoten der Wert eins zurückgegeben und schlussendlich die entsprechenden Anzahlen der Communities in der Rekursion nach oben durchgegeben. Die Anzahl an Autoren in einer Community wird jeweils gespeichert, um diese Werte in die Datenbank schreiben zu können.

4.4 Conductance

Die Conductance einer Community ist durch folgende Formel [**?**] bestimmt:

$$\frac{c(C,G\backslash C)}{min(k_C,k_{G\backslash C})}$$

Dabei haben die Werte die folgende Bedeutung:

- C: Community
- G: Gesamtgraph
- c(C, G \C): Cut size von C
- k_C: Anzahl Kanten von C
- $k_{G\backslash C}$: Anzahl Kanten von G ohne C

Somit ist die Conductance ein Verhältnis der Publikations-Kanten ,welche aus der Community heraus gehen zu der Anzahl an Publikations-Kanten innerhalb der Community.

Der erwartete Wertebereich der Conductance liegt zwischen Null und Eins. Dabei würde Null bedeuten, dass keine Kanten nach außen gehen und die Community komplett isoliert ist. Ein Wert größer Eins würde bedeuten, dass die Community mehr Kanten nach außen hat, als innere Kanten. Da der Community Algorithmus zur Erkennung von Communities die Anzahl der Kanten zwischen den Knoten betrachtet, ist es eher unwahrscheinlich, dass eine Community mit einer Conductance größer Eins entsteht.

4.5 Ergebnisse in die Datenbank schreiben

Da sehr viele Knoten und Kanten in der Datenbank gespeichert sind und alle Messergebnisse zu den jeweiligen Knoten und Kanten in die Datenbank geschrieben werden sollen, wird eine effektive Möglichkeit benötigt die große Menge an Messergebnissen in die Datenbank zu schreiben.

4.6 Ergebnisse der Messungen

4.7 Größe

4.8 Conductance

4.9 Analyse der Ergebnisse

5 Autoren

Durch die im Abschnitt **??**Community-Detection) gefundenen Communities kann man jetzt auch die Beziehung eines Autors zu der ihm zugeordneten Community betrachten.

In diesem Abschnitt wird untersucht ob verschiedene Rollenverteilungen unter den Autoren einer Community erkennbar sind. Hierbei wird festgestellt welche Rollen unter den Autoren auftreten und welche Rolle etwa, die Autoren einnehmen die am meisten publizieren.

Dazu wurden die in [**?**] beschriebene Klassifizierung von Knoten verwendet. Um die Knoten klassifizieren zu können muss jeweils die With-in-module-degree (z-score) und für sie und ihre Communities berechnet werden. Anschließend kann die Klassifizierung des Knotens wie in Abbildung imager1-r7 beschrieben erfolgen. Grundsätzlich wird in diesem Model zwischen folgenden Rollen unterschieden:

- Nabenknoten (hubs)

  (R5) provincial hubs (provinzielle Naben/Zentren)

  (R6) connector hubs (verbindende Naben/Zentren)

  (R7) kinless hubs (sippenlose Naben/Zentren)

- Einzelknoten (non-hubs)

  (R1) ultraperipheral (sehr dezentral)

  (R2) peripheral (dezentral)

  (R3) connectors (Bindeglieder)

  (R4) kinless vertices (Einzelknoten)

Zur Klassifizierung der Autor-Knoten im Co-Autor-Graphen, wird für jeden Knoten die Community betrachtet in der er sich befindet (Top-Level-Community).

6 Schluss

Durch eine modulare Architektur, ist es einfach neue Messmodule in das System zu integrieren und Ergebnisse in die Datenbank zu schreiben. Dadurch ist die Infrastruktur beliebig erweiterbar. Es könnten neue Messwerte wie beispielsweise die *Significance* (Stabilität einer Community) eingeführt werden, um noch mehr über die Daten aussagen zu können. Auch wäre eine Betrachtung der zeitlichen Entwicklung von Communities und Autoren möglich.

Weiterhin wäre es möglich neue Parser zu entwickeln, um die Messungen auch auf andere Daten durchzuführen.

Abkürzungsverzeichnis

DBLP DataBase systems and Logic Programming

Literatur