Effiziente Suche in semantischen Netzen

Efficient Querying in Semantic Networks

Neumann, Thomas; Theobald, Martin

Max-Planck-Institut für Informatik, Saarbrücken

Korrespondierender Autor/in

E-Mail: neumann@mpi-inf.mpg.de

Zusammenfassung

Daten, die eine Netz- oder Graphstruktur aufweisen, sind heute allgegenwärtig, angefangen von sozialen Netzen im Internet bis hin zu komplexen biologischen Netzen. Die Kombination von Netzstruktur und zugehöriger Semantik stellt besondere Anforderungen an Suche und automatisches Schließen. In unserer Gruppe haben wir deshalb zwei Systeme für große Datengraphen entwickelt: RDF-3X, das auf die effiziente Suche in klassischen Datengraphen spezialisiert ist, und URDF, das Suche und Schließen unter Untersicherheit unterstützt.

Summary

Data that exhibits a network or graph structure is ubiquitous, ranging from social networks on the internet to complex networks in life sciences. The combination of network structure within the data and an associated semantic imposes great challenges to searching and automatic reasoning. Our group therefore developed two system for managing large data graphs: RDF-3X, which aims for efficient query processing in classical data graphs, and URDF, which supports searching and reasoning under uncertainty.

Einführung

Die meisten Dinge haben Beziehungen zu anderen Dingen. Ein Buch hat beispielsweise einen oder mehrere Autoren, ein Protein nimmt an bestimmten Reaktionen teil, und ein Benutzer von Web 2.0 Webseiten hat Verknüpfungen zu seinen Freunden. Diese Beziehungen zwischen Dingen, oder abstrakter: Entitäten bilden zusammen mit den Entitäten selbst eine Netz- oder Graphstruktur. Durch eine zugehörige formale Semantik werden die Datengraphen zu semantischen Netzen, d. h. Netzen in denen automatisiert Schlüsse gezogen werden können. Dabei ist es häufig so, dass die Beziehungen selbst interessanter sind als die eigentlichen Entitäten. Bei sozialen Interaktionsgraphen beispielsweise interessiert man sich für die besonders aktiven Nutzer, bei biologischen Netzen interessiert man sich für die Stoffwechselwege, die durch die Interaktionen der Proteine bestimmt werden, usw.

Die zugrundeliegenden Datengraphen können dabei sehr groß werden; die UniProt Datensammlung über Proteininformationen enthält beispielsweise knapp eine Milliarde Verknüpfungen. Auf solch großen Daten

komplexe Analysen oder Inferenzen durchzuführen führt schnell zu komplexen Optimierungsproblemen und erfordert hocheffiziente Algorithmen für Datenhaltung und Suche.

RDF-3X – Effiziente Anfrageverarbeitung auf RDF-Daten

Ein Datenformat, das speziell für graphstrukturierte Daten entworfen wurde, ist das Resource Description Framework (RDF). RDF wurde 2004 vom World-Wide-Web-Consortium (W3C) veröffentlicht und beschreibt zunächst einmal auf syntaktischer Ebene ein Format für semantische Netze sowie die zugehörige formale Semantik. Entworfen wurde RDF ursprünglich als Teil der Semantic-Web-Initiative, aber heute wird das Datenformat sehr vielfältig eingesetzt.

Beispielgraph für RDF-Daten.

Abbildung 1 zeigt ein Beispiel für graphstrukturierte Daten über Bücher und ihre Autoren. Eine RDF-Datensammlung besteht aus *Tripeln*, die jeweils einer Kante und dem zugehörigen Knotenpaar im Graphen entsprechen. Ein Tripel besteht in RDF-Schreibweise aus *Subjekt, Prädikat* und *Objekt*; im Graph entspricht dies der mit *Prädikat* beschrifteten Kante von *Subjekt* zu *Objekt*. Im Beispiel gibt es die Kante , die ausdrückt, dass das Subjekt "id1" (mit dem Namen Arthur Conan Doyle) 1859 geboren wurde. Weitere Kanten wie , und geben an, dass das Objekt "id2" (mit dem Titel „Das Zeichen der Vier“) von id1 geschrieben wurde, dass id1 in Edinburgh geboren ist und dass Edinburgh in Schottland liegt. Diese Kanten im Graphen entsprechen somit den Subjekt-Prädikat-Objekt-Tripeln:

, , , , usw.

Auch komplexe Zusammenhänge lassen sich durch diese Art von Verweisen relativ einfach formulieren. Die Einfachheit und Flexibilität, die durch die Graphstruktur der Daten erreicht wird, führt allerdings dazu, dass die Suche in RDF relativ aufwendig ist. Suchanfragen werden normalerweise in einer Anfragesprache wie z. B. SPARQL formuliert und beschreiben ein Muster, das in den Daten gesucht werden soll. Wenn man beispielsweise die Titel von Büchern schottischer Autoren sucht, kann man das mit folgenden Tripel-Mustern beschreiben:

?autor geborenIn ?stadt, ?stadt liegtIn Schottland, ?autor autorVon ?buch, ?buch Titel ?titel

Die mit Fragezeichen beginnenden Teile sind jeweils Variablen, deren Werte vom Datenbanksystem bestimmt werden müssen; die Teile in spitzen Klammern sind vom Benutzer als Suchbedingung vorgegebene Werte. Durch das mehrfache Verwenden von Variablen wie ?autor werden die einzelnen Tripel-Muster verknüpft und der globale Zusammenhang hergestellt. Im Anfrageergebnis müssen alle Vorkommen der gleichen Variable an den gleichen Wert gebunden sein, was implizit einen Verbund der einzelnen Muster erzeugt. Für die Anfrage

hier heißt das beispielsweise, dass man zunächst eine Liste aller Städte in Schottland erzeugen kann (was Bindungen für ?stadt erzeugt) und dann eine Liste mit allen Geburtsorten von Autoren (was andere Bindungen für ?stadt erzeugt). Anschließend müssen beide Listen verbunden werden, und dabei werden nur diejenigen Bindungen von ?stadt erhalten, die in beiden Listen vorkommen.

Grundsätzlich könnte diese Anfrage auf verschiedene Arten ausgewertet werden. Man könnte zum Beispiel zunächst alle Städte in Schottland bestimmen, dann alle Autoren heraussuchen, die in einer dieser Städte geboren wurden, deren Bücher bestimmen, und dann die Titel ausgeben. In den meisten Fällen ist das wahrscheinlich auch die beste Strategie. Falls es in der Datenbasis aber nur wenige Autoren, dafür aber sehr viele Städte gibt, könnte es allerdings besser sein, zunächst die Geburtsorte aller Autoren zu bestimmen, dann zu prüfen, ob der Ort in Schottland liegt, und dann die Bücher mit Titeln herauszusuchen. Dieses Abwägen zwischen verschiedenen Ausführungsalternativen ist eine wesentliche Herausforderung bei der Verarbeitung von RDF-Daten. Eine RDF-Datensammlung kann viele Millionen oder sogar Milliarden von Tripeln bzw. Kanten im Graph enthalten; bei solchen Datenmengen wird sowohl die Auswahl einer hinsichtlich Laufzeit günstigen Ausführungsalternative als auch die eigentliche Suche an sich sehr komplex.
Das wird schon deutlich wenn man allein die Anzahl der Ausführungsalternativen betrachtet. Bereits bei 10 Tripel-Mustern ergben sich mehr als 17 Milliarden Alternativen, was auf heutigen Rechnern nicht in akzeptabler Zeit durchgerechnet werden kann. Eine effiziente Suche in komplexen RDF-Graphen erfordert deshalb raffiniertere Optimierungstechniken [1].

Am Max-Planck-Institut für Informatik (MPI-INF) haben wir das Datenbanksystem RDF-3X (RDF Triples Express) [2] für die effiziente Verarbeitung von RDF-Daten entwickelt, das diese Probleme auf mehreren Ebenen angeht. Zunächst einmal muss man die Daten selbst geeignet speichern, damit einzelne Tripel-Muster effizient ausgewertet werden können. Wir haben dazu die Daten geschickt komprimiert und mit Suchbäumen indexiert, so dass jedes beliebige Tripel-Muster sehr schnell ausgewertet werden kann. Das alleine reicht jedoch nicht aus, denn Benutzer sind meistens an größeren Zusammenhängen interessiert und stellen deshalb Anfragen mit mehreren verknüpften Tripel-Mustern. Wir bilden deshalb solche zusammengesetzten Anfragen auf Ausführungsstrategien mit algebraischen Operatoren ab und wählen aus den verschiedenen Ausführungsalternativen diejenige aus, die voraussichtlich am schnellsten die Antworten zur Anfrage liefern kann. Die Laufzeitprognose für verschiedene Alternativen benötigt Statistiken und enstprechende Schätzverfahren. Für die Beispielanfrage müsste man etwa schätzen, ob es mehr Autoren oder mehr Städte gibt und wie sich dies auf die Laufzeit der Ausführung auswirkt. Eine sorgfältige Auswahl der Ausführungsstrategie kann die Anfrageverarbeitung oft um einen Faktor von 10 oder mehr beschleunigen. Ein wesentlicher Faktor ist dabei die effiziente Bestimmung der besten Ausführungsstrategie. Während der gesamte Suchraum der Ausführungsalternativen zu groß für eine vollständige Suche ist, beschränkt RDF-3X die Suche auf erfolgsversprechende Alternativen und reduziert durch effiziente Algorithmen die Anzahl der betrachteten Alternativen bei typischen Anfragen auf einige tausend. Mithilfe von statistischen Modellen werden diese Alternativen bewertet und dann die effizienteste ausgewählt.

Dieses Zusammenspiel von effizienter Datenhaltung und darauf aufbauender optimierter Anfrageverarbeitung erlaubt es, auch sehr große RDF-Datenmengen effizient zu durchsuchen. Die Datensammlung UniProt, die größte bioinformatische Datenbank über Proteine, enthält beispielsweise über 850 Million Tripel bzw. Kanten. Dennoch lassen sich in RDF-3X auch komplexe Anfragen typischerweise in weniger als einer Sekunde beantworten. Das RDF-3X System ist open source und frei verfügbar (http://www.mpi-inf.mpg.de/~neumann/rdf3x).

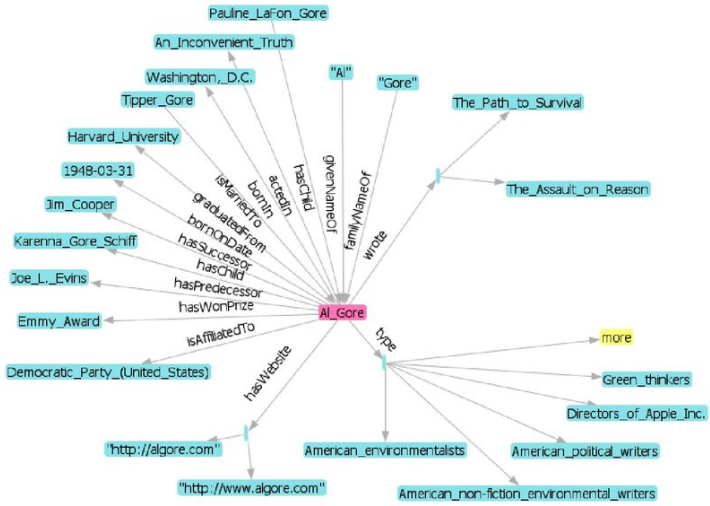
Automatische Wissensextraktion aus Web-Dokumenten

Die weltweite Forschung im Bereich der automatischen Wissensextraktion hat in den letzten Jahren enorme Fortschritte erzielt. Zu aktuellen Forschungsprojekten in diesem Bereich zählen unter anderem Dbpedia [3], Intelligence-In-Wikipedia, KnowItAll oder Cimple, sowie die am Max-Planck-Institut für Informatik entwickelte YAGO [4] Wissensbasis, die mit mehr als 20 Millionen extrahierten RDF-Fakten eine der größten automatisch extrahierten und frei verfügbaren Wissensbasen darstellt. Dabei stehen insbesondere frei verfügbare Enzyklopädien wie Wikipedia im Fokus der automatischen Wissensextraktion, da diese auf sehr reichhaltige und allgemeinverständliche Art widerspiegeln, wie Millionen von Menschen auf kollaborative Weise Informationen und Wissen bearbeiten bzw. repräsentieren.

Allerdings liegt es sowohl in der kollaborativen Natur dieser Enzyklopädien als auch an der immer noch beträchtlichen Fehleranfälligkeit der maschinellen Informationsextraktion, dass automatisch extrahierte RDF-Wissensbasen ein teilweise beträchtliches Maß an Unschärfe oder sogar Inkonsistenzen enthalten können. So können zum Beispiel ungenaue Angaben eines Autors eines solchen Artikels in die Wissensbasis übernommen werden, oder es können semantische Disambiguierungsprobleme zu falschen Zuordnungen führen, was zu Inkonsistenzen in der Wissensrepräsentation oder beim späteren Anfrageauswerten führen kann. Eine ähnliche Beobachtung gilt darüber hinaus auch für Wissensbasen, die durch andere automatisierte Verfahren [5] wie dem Verschmelzen unterschiedlicher ontologischer Quellen oder der Integration von benutzergenerierten Inhalten erstellt wurden. Interessanterweise weisen sogar sorgfältig von Hand gepflegte Wissensbasen wie SUMO bei genauer Validierung subtile Inkonsistenzen auf, was die enorme Schwierigkeit der Generierung einer formal korrekten, also im logischen Sinne vollständig validierbaren, Wissensbasis zeigt.

Unvollständigkeit, Ungenauigkeit und Inkonsistenz

Trotz der Fortschritte der letzten Jahre bleibt es eher unwahrscheinlich, dass eine Wissensbasis, die automatisch aus einer Web-Enzyklopädie wie Wikipedia extrahiert wurde, jemals alle in dieser enthaltenen Informationen mit perfekter Präzision widerspiegeln und in einem strukturierten (also maschinenlesbaren) Format wie RDF repräsentieren kann. Informationsextraktion aus Webdokumenten bleibt daher auch zwangsläufig unvollständig. So könnte ein Benutzer zum Beispiel die scheinbar einfache Anfrage *„Wo wohnt Al Gore?*" stellen – in SPARQL: *Al_Gore ?place*. **Abbildung 2** zeigt einen Ausschnitt der am MPI-INF extrahierten Wissensbasis, YAGO [4], mit allen in der Wissensbasis enthaltenen Fakten über *Al Gore*. Obwohl YAGO als eine der umfassendsten aus Wikipedia extrahierten Wissensbasen gilt, ist der gesuchte Fakt über den Wohnort von Al Gore nicht enthalten.



Yago-Fakten über Al Gore.

Erst bei genauerer Analyse des ursprünglichen Wikipedia-Artikels über Al Gore finden wir den Grund: Sogar einem Menschen erschließt sich die Antwort nur schwer aus dem Artikel, einfach weil es keinen genauen Hinweis auf den gesuchten Wohnort in diesem Artikel gibt. Erst mithilfe von *probabilistischen oder heuristischen Regeln* können wir auf den gesuchten Wohnort, in diesem Falle Washington D.C., schließen, da wir dem Artikel (und der resultierenden Wissensbasis) entnehmen können, dass Al Gore verheiratet ist, und dass seine Ehefrau, Tipper Gore, in Washington D.C. wohnt. Die Anwendung einer solchen Regel wird im Allgemeinen als *Inferenz* bezeichnet, wobei als Ergebnis einer solchen Inferenz auch auf Fakten geschlossen werden kann, die nicht in den ursprünglich extrahierten Daten enthalten sind. Automatische Inferenztechniken können uns also dabei helfen, Unvollständigkeiten in der Wissensrepräsentation zu kompensieren.

Darüber hinaus sind automatisch extrahierte Daten aber auch häufig *inkorrekt*. So könnte ein Extraktor zum Beispiel in 90 Prozent der Fälle ein Geburtsdatum korrekt erkennen, aber nur in 60 Prozent der Fälle einen Personennamen korrekt von einem Firmennamen unterscheiden können. Verschiedene automatische Extraktionsverfahren liefern daher *Konfidenzwerte* für Fakten, also (geschätzte) Wahrscheinlichkeiten für die Korrektheit der extrahierten Fakten, die sich ebenso in den Konfidenzen für die Korrektheit der Anfrageantworten widerspiegeln sollten. Wenn wir also nach Personen und deren Geburtsdaten suchen, brauchen wir ein probabilistisches Modell, das die Eingabekonfidenzen der für die Antwort benötigten Fakten mit ihrer logischen Verknüpfung in der Antwort korrekt widerspiegelt.

Als dritte Form von Unschärfe kann automatische Informationsextraktion aus Webdokumenten schließlich sogar zu beträchtlichen *Inkonsistenzen* in der Wissensrepräsentation führen. So könnten beispielsweise im Extraktionsschritt zwei unterschiedliche Fakten *bornIn*(Al_Gore,Washington_D.C.) und *bornIn*(Al_Gore,New_York_City) aus unterschiedlichen Quellen für den Geburtsort von Al Gore in die Wissensbasis eingefügt worden sein. Je nach Auswertungsstrategie würden strikte, also rein logikbasierte, Verfahren daher oft eine leere Antwortmenge liefern, oder es könnten sich – was noch schlimmer wäre – beliebige Antworten aus einem einzigen Widerspruch in der Wissensbasis ableiten lassen („ex falso quod libet").

URDF – Automatisches Schließen in unsicheren Wissensbasen

Oft sind solche Inkonsistenzen allerdings nicht so leicht zu entdecken wie im obigen Beispiel und sogar einem menschlichen Benutzer nicht auf den ersten Blick ersichtlich. In großen Wissensbasen mit Millionen von Fakten

kann eine automatisierte Suche nach Inkonsistenzen häufig nur durch komplizierte Inferenzschritte ausgeführt werden, wobei automatisierte Verfahren zum Erkennen und Entfernen von Inkonsistenzen sogar zum Verlust korrekter Daten führen können. Automatisches Schließen in großen Wissensbasen braucht daher eine hohe Robustheit im Hinblick auf unvollständige, ungenaue oder sogar inkonsistente Daten, welche wir im Folgenden unter dem Begriff *unsichere* Daten bezeichnen wollen. Diese Unsicherheit verlangt nach neuen – relaxierten aber dennoch effizienten – Auswertungsstrategien für SPARQL, die Standardanfragesprache für in RDF gespeicherte Wissensbasen, die auch unsichere Daten dynamisch, also *während der Anfrageauswertung*, handhaben können. Während RDF-3X auf sehr effiziente Weise mit exakten Daten umgehen kann, fokussiert sich die Forschung in unserem neuesten Projekt, URDF, daher speziell auf die Anfragebearbeitung mit unsicheren Daten. URDF unterstützt dabei den internationalen Web Ontology Language Standard, OWL-lite, für semantische Wissensrepräsentationen und vereint sowohl regelbasierte also auch probabilistische Inferenzmechanismen, wie sie uns im obigen Beispiel auf der Suche nach dem möglichen Wohnort von Al Gore geholfen haben. Die enorme Expressivität der regelbasierten und probabilistischen Inferenztechniken bringt natürlich auch enorme Herausforderungen an neue, effiziente Auswertungsstrategien für SPARQL-Anfragen mit sich, die wir mit URDF verfolgen. Unsere aktuellen Experimente auf großen Wissensbasen mit mehr als 20 Millionen Fakten und vielen verschachtelten Inferenzregeln zeigen bereits interaktive Antwortzeiten im Sekundenbereich, was eine bisher unerreichte Effizienz und Skalierbarkeit für logikbasierte Inferenzmechanismen darstellt.

Originalveröffentlichungen

Nach Erweiterungen suchenAbsatzBilderweiterungDateilisteHTML-ErweiterungJobtickerKalendererweiterungLinkerweiterungMPG.PuRe-ReferenzMitarbeiter (Employee Editor)MitarbeiterlistenerweiterungPersonenerweiterungPublikationserweiterungRSS tickerTaglistenerweiterungTeaser mit BildTextblockerweiterungVeranstaltungstickererweiterungVideoerweiterungYouTube-Erweiterung

[1] **T. Neumann, G. Weikum:**

Scalable join processing on very large RDF graphs.
SIGMOD Conference, 627–640 (2009).

[2] **T. Neumann, G. Weikum:**

The RDF-3X engine for scalable management of RDF data.
VLDB Journal **19**, 91–113 (2010).

[3] **C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, S. Hellmann:**

DBpedia - A crystallization point for the Web of Data.
Journal of Web Semantics **7**, 154–165 (2009).

[4] **F. M. Suchanek, G. Kasneci, G. Weikum:**

Yago: a core of semantic knowledge.
World Wide Web Conference, 697–706 (2007).

[5] **G. Weikum, G. Kasneci, M. Ramanath, F. M. Suchanek:**

Database and information-retrieval methods for knowledge discovery.
Communications of the ACM **52**, 56–64 (2009).