DIPLOMARBEIT

Integration von Daten
aus sozialen Online-Netzwerken

von

Andreas Kurz

eingereicht am 09.04.2008 beim
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren (AIFB)
der Universität Karlsruhe

Referent: Prof. Dr. Studer
Betreuer: Dipl. Inform. Max Völkel

Heimatanschrift:
Eva-Maria-Buch-Str. 31
76189 Karlsruhe
andr.kurz@gmail.com

Studienanschrift:
Eva-Maria-Buch-Str. 31
76189 Karlsruhe

Erklärung:

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, 9. April 2008

Inhaltsverzeichnis

1 Einleitung

Seit einigen Jahren sind soziale Online-Netzwerke sehr beliebt. Durch die jederzeitige Erreichbarkeit von beliebigen internetfähigen Geräten mit Internetanschluss vereinfachen sie die Pflege bestehender Kontakte und die Knüpfung neuer. Die Weise, auf die dies geschieht, ist in jedem Online-Netzwerk ähnlich. Man legt ein Profil an, gibt Informationen über sich ein und erstellt Kontakte zu anderen Benutzern, die man dafür im echten Leben nicht kennen muss.

Ein soziales Online-Netzwerk ist ein Beziehungsgeflecht im Internet. Es dient dem Austausch der persönlichen Daten, dem Herstellen und dem Vertiefen der Beziehungen zwischen den Teilnehmern.

Laut comScore[1] besuchten im Juli 2007 45 Prozent der deutschen Internetnutzer soziale Online-Netzwerke. Dabei war MySpace[2] das beliebteste Ziel. Soziale Online-Netzwerke unterscheiden sich teilweise sehr stark in ihren Zielgruppen. So gibt es Netzwerke, die die Knüpfung und Pflege der beruflichen Kontakte als Schwerpunkt haben. Eines der größten solcher Netzwerke ist Xing[3]. Ein anderes Beispiel für ein Netzwerk, das eine besondere Zielgruppe hat, ist StudiVZ[4]. Es richtet sich an Studenten und Hochschulmitarbeiter. Trotz der Tatsache, dass viele soziale Online-Netzwerke unterschiedliche Zielgruppen haben, sind in verschiedenen Netzwerken oft die gleichen Informationen über die Nutzer enthalten. Diese Informationen sind zum Beispiel Vor- und Nachname, Wohnort, Email-Adresse.

1.1 Probleme

Mit den Anmeldungen eines einzelnen Nutzers in mehreren Netzwerken ergeben sich einige Probleme für den Nutzer. Der Verwaltungs- und Pflegeaufwand mehrerer Nutzerkonten nehmen viel Zeit in Anspruch. Außerdem bietet die in den sozialen Online-Netzwerken vorhandene Datenmenge eine Möglichkeit für Netzwerkforscher, verschiedene Analysen auf diesen Daten durchzuführen [WFI94].

Problem 1: Die Pflege bestehender und der Aufbau neuer Kontakte spielen in den sozialen Online-Netzwerken eine wichtige Rolle. Um eine neue Person in den persönlichen sozialen Online-Netzwerken in seine Kontaktlisten eintragen zu können, muss in jedem Netzwerk einzeln die Suchfunktion verwendet werden.

[1] http://www.comscore.com
[2] http://www.myspace.com
[3] http://www.xing.com
[4] http://www.studivz.net

Problem 2: Durch die vielen verschiedenen Nutzerkonten kann man den Überblick verlieren, in welchem Netzwerk man welche Kontakte hat. Das erschwert das Auffinden der gesuchten Informationen, zum Beispiel die aktuelle Telefonnummer oder Adresse eines Bekannten.

Problem 3: Der Nutzen eines speziellen sozialen Online-Netzwerkes ist für den Nutzer nicht ersichtlich. Der Nutzer kann redundante Kontakte zu gleichen Personen in veschiedenen Netzwerken haben.

Problem 4: Viele Nutzer verwenden Adressbücher wie zum Beispiel Outlook. Da die meisten sozialen Online-Netzwerke keine oder eingeschränkte Möglichkeiten bieten, seine Daten und die Daten seiner Kontakte lokal abzuspeichern, fehlt für die Nutzer die Möglichkeit die Daten aus den Netzwerken mit seinem Adressbuch abzugleichen.

Für Xing gibt es eine Software[5], die die Kontaktdaten der eigenen Freunde mit seinem digitalen Adressbuch, zum Beispiel Outlook, abgleichen kann.

Problem 5: Viele soziale Online-Netzwerke bieten nur beschränkte Analysemöglichkeiten des eigenen Profils. Zum Beispiel kennt der Nutzer nicht das 'soziale Kapital' seines Profils.

Unter sozialem Kapital ist hier die Gesamtheit der aktuellen und potenzieller Ressourcen, die aus den Beziehungen im Netz resultieren, zu verstehen. Durch die Bewertung des sozialen Kapitals erhält der Nutzer dann praktisch die Wichtigkeit der einzelnen Netzwerke und seiner Kontakte für sich selbst. Außerdem ist die Bewertung des eigenen Profils mit den Bewertungen der Profile der Anderen vergleichbar [Bou92].

Problem 6: Da die meisten sozialen Online-Netzwerke kein Interesse haben, die Daten aus ihren Netzwerken für Analysen zur Verfügung zu stellen, fehlt Netzwerkforschern die Möglichkeit, unterschiedliche Fragestellungen zu beatworten. Einige einfache exemplarische Fragestellungen sind:

- Woher kommen die Teilnehmer, woher kommen ihre Freunde?
- Wie stark sind die Teilnehmer verlinkt, wieviele Kontakte haben sie im Durchschnitt? Gibt es diesbezüglich Unterschiede in verschiedenen Netzwerken?
- Wieviel Prozent der Teilnehmer geben bestimmte Daten frei, wieviel Prozent geben überhaupt Daten frei?

[5] http://www.eworks.de/references/xing-plugin.html

- Untersuchung der Teilnetzwerke: Wie oft kommt es vor, dass man die gleichen Freunde wie seine Freunde hat?
- Gibt es Korrelationen zwischen Anmeldungen in bestimmten Netzwerken, wie oft sind Teilnehmer aus Netzwerk A auch in Netzwerk B angemeldet?
- Wie sieht die Verteilung der Altersstruktur, des Bildungsstandes der Teilnehmer aus?

1.2 Idee

Die Idee ist der Entwurf und die Implementierung eines Systems, das verschiedene nutzerrelevante Daten aus unterschiedlichen sozialen Online-Netzwerken extrahieren und aggregieren kann (siehe Abbildung 1). Um die Aggregation durchzuführen, wird ein einheitliches Datenmodell, SNEQ[6]-Datenmodell, entworfen, das alle Daten aus sozialen Online-Netzwerken aufnehmen kann. Das System hilft, sowohl für die einzelnen Nutzer als auch für die Netzwerkforscher, die in vorherigem Abschnitt beschriebenen Probleme zu lösen.

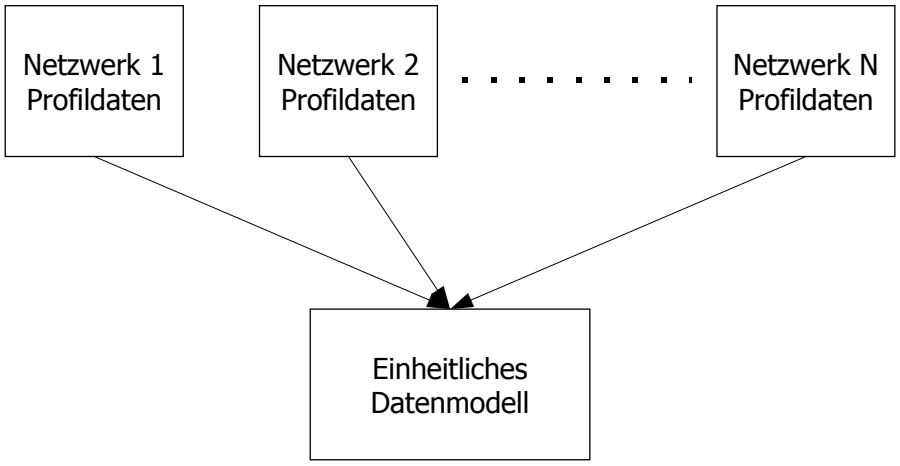

Abbildung 1: Zentrale Idee - Extraktion und Aggregation

1.3 Gliederung

In Kapitel 2 werden einige der beliebtesten sozialen Online-Netzwerke kurz vorgestellt. Es werden Grundlagen des Semantic Webs und der Duplikater-

[6]Social Network Export and Query

kennung vorgestellt und erläutert, auf denen diese Arbeit aufbaut. Es folgt eine Vorstellung verwandter Arbeiten und einiger bekannter Schnittstellen von sozialen Online-Netzwerken.

Kapitel 3 beschreibt den Kern dieser Arbeit. Im ersten Teil werden die Auswahl und die Analyse von drei sozialen Online-Netzwerken und der Entwurf eines einheitlichen Datenmodells beschrieben. Das Datenmodell kann alle nutzerrelevanten Daten aus den ausgewählten Netzwerken aufnehmen. Im zweiten Teil wird ein System entworfen, das die Daten aus den ausgewählten Netzwerken extrahieren und sie im entworfenen einheitlichen Datenmodell speichern soll. Außerdem soll das System Profile von gleichen Personen aus unterschiedlichen Netzwerken erkennen.

Kapitel 4 beschreibt die Implementierung des im Kapitel 3 entworfenen Systems.

In Kapitel 5 wird die Evaluation durchgeführt. Die Evaluation wird für die im Abschnitt 3.2 aufgestellten Systemanforderungen und für die im Abschnitt 1.1 beschriebenen Probleme durchgeführt.

Kapitel 6 enthält eine Zusammenfassung dieser Arbeit und beschreibt einen Ausblick auf mögliche zukünftige Entwicklungen.

2 Grundlagen und verwandte Arbeiten

Dieses Kapitel enthält eine Vorstellung der beliebtesten sozialen Online-Netzwerke, der Grundlagen, auf den diese Arbeit aufbaut, und der verwandten Arbeiten.

2.1 Soziale Online-Netwerke

In diesem Abschnitt werden alphabetisch die nach Kontinenten populärsten[7] sozialen Online-Netzwerke kurz vorgestellt.

Bebo[8] ist vor allem in Irland und Großbritannien bekannt. Es hat etwa 25 Mio Mitglieder. Außer der Kommunikation nutzen die Teilnehmer das Netzwerk zum Hochladen von Bildern und zum Schreiben von Blogs. Im März 2008 wurde Bebo vom Time Warner-Konzern über AOL übernommen[9].

CyWorld[10] ist sehr populär in Südkorea. Die meisten Mitglieder sind Südkoreaner. Besonders erstaunlich ist die Tatsache, dass jeder vierte Südkoreaner dort angemeldet ist. CyWorld hat über 20 Mio Mitglieder.

Facebook[11] war zuerst nur für Studenten der Harvard Universität geöffnet. Später öffnete sich Facebook für alle Studenten. Mittlerweile kann sich jederman dort anmelden. Eine Besonderheit in Facebook ist, dass die Teilnehmer sogenannten Networks beitreten können. Alle Teilnehmer eines Networks habes eine bestimmte Gemeinsamkeit. Zum Beispiel haben sie alle den gleichen Arbeitgeber oder sind an der gleichen Universität. Auch gibt es die Möglichkeit, Bilder hochzuladen und sie anderen Netzwerkteilnehmern zu zeigen. Laut Alexa.com[12] belegt Facebook Platz 7 bei den Seitenaufrufen aus der ganzen Welt. Bei den Aufrufen aus den USA ist Facebook auf Platz 5 und aus Großbritannien auf Platz 2. Aus diesen Statistiken und aus der Tatsache, dass es lange Zeit nur eine englische Benutzerschnittstelle gab, resultiert, dass Facebook vor allem im englischsprachigen Raum sehr beliebt ist. Facebook hat etwa 50 Mio Mitglieder. Eine deutsche Benutzerschnittstelle wurde angekündigt und kann seit April 2008 verwendet werden.

[7] http://www.lemonde.fr/web/infog/0,47-0@2-651865,54-999097@51-999297,0.html
[8] http://www.bebo.com
[9] http://www.ireland.com/newspaper/finance/2008/0314/1205104771653.html
[10] http://www.cyworld.com/main2/index.htm
[11] http://www.facebook.com
[12] http://www.alexa.com

Friendster[13] war bis April 2004 gemäß den Seitenzugriffen das meist besuchte soziale Online-Netzwerk. Vor allem ist Friendster im englischsprachigen und asiatischen Raum bekannt. Friendster hat 50 Mio Mitglieder.

Hi5[14] hat etwa 98 Mio Nutzer. Sehr beliebt ist es in Lateinamerika. Laut Alexa.com belegt Hi5 Platz 6 der meisten Seitenzugriffe aus Mexiko.

LinkedIn[15] ist ein berufsorientiertes soziales Online-Netzwerk. Kontakte sollen eigene Geschäftsbeziehungen widerspiegeln, wen man kennt und wem man vertraut. LinkedIn hat etwa 19 Mio. Mitglieder. Eine Besonderheit von LinkedIn ist, dass die Arbeitgeber freie Stellen bekanntmachen und nach geeigneten Kandidaten suchen können.

LiveJournal[16] ist eine Blogseite mit den Funktionen von sozialen Online-Netzwerken. Es hat etwa 15 Mio Mitglieder. US-amerikanische Mitglieder stellen die Mehrheit im Netzwerk dar. LiveJournal hat eine relativ große Beliebtheit in Russland.

MySpace[17] ist mit 180 Mio Nutzern immernoch das zurzeit größte soziale Online-Netzwerk. Ursprünglich war MySpace ein Dienst zur kostenlosen Speicherung der Daten im Internet. Seit der Gründung liegt ein Schwerpunkt von MySpace auf Musik. Künstler haben die Möglichkeit, zu ihren Fans Kontakt zu halten. Die Datenspeicherung ermöglicht es den Künstlern, Hörproben in MySpace einzustellen. Mittlerweile ist es auch möglich, Filme anzubieten, was insbesondere Filmemacher interessieren könnte. Die Bedeutung von MySpace wird anhand von globalen Seitenzugriffen deutlich. Laut Alexa.com ist MySpace auf Platz 5 der weltweiten Internetseitenaufrufe. Betrachtet man nur die Aufrufe aus den USA, liegt MySpace sogar auf Platz 3. Bei Aufrufen aus Großbritannien belegt MySpace Platz 9, bei denen aus Deutschland Platz 11. Diese Statistiken zeigen auch, dass der Großteil der MySpace-Nutzer aus dem nordamerikanischen englischsprachigen Raum kommt.

Netlog[18] hat nach eigenen Angaben etwa 33 Mio Mitglieder. Die Benutzerschnittstelle von Netlog unterstützt 15 Sprachen.

[13] http://www.friendster.com
[14] http://hi5.com
[15] http://www.linkedin.com
[16] http://www.livejournal.com
[17] http://www.myspace.com
[18] http://corporate.netlog.com

Ning[19] ist eine soziale Online-Plattform. Eine Besonderheit von Ning ist, dass die Nutzer ihre eigene Netzwerke eröffnen können. Ning verwaltet momentan über 185000 Netzwerke[20].

Orkut[21] wird von Google betrieben. Im Moment hat Orkut etwa 120 Mio Mitglieder. Sehr beliebt ist Orkut vor allem in Brasilien und Indien. Ursprünglich konnte man Orkut nur beitreten, wenn man eine Einladung hatte. Eine weitere Besonderheit von Orkut ist, dass man eine Bewertung seiner Kontakte abgibt. Mögliche Bewertungen sind zum Beispiel 'Beste Freunde', 'Bekanntschaft' oder 'Nie getroffen'.

Plaxo[22] ist spezialisiert auf Online-Adressverwaltung. Es gibt ein Plugin für alle gängigen Email- und Adress-Programme wie Microsoft Outlook, Mozilla Thunderbird und Mac OS Adressbuch. Plaxo hat 15 Mio Mitglieder.

Skyblog[23] hat etwa 21 Mio Mitglieder. Es war anfangs eine reine Blogseite. Seit Anfang 2007 wurden Funktionen der sozialen Online-Netzwerke hinzugefügt. Skyblog hat die größte Popularität im französischsprachigen Raum.

StudiVZ[24] ist ein deutschsprachiges soziales Online-Netzwerk. Es ist im Wesentlichen eine Nachbildung von Facebook. StudiVZ hat etwa 4 Mio Nutzer. Die Nutzer kommen zum größten Teil aus Deutschland, Österreich und der Schweiz. Das liegt an der ausschließlich deutschsprachigen Benutzerschnittstelle. Zwar wurde StudiVZ für Studenten konzipiert, kann aber von jederman benutzt werden. Mittlerweile sind mit SchülerVZ und MeinVZ zwei Ableger verfügbar, die auf die Zielgruppen Schüler und Erwachsene, die nicht notwendigerweise mit einer Hochschule in Verbindung stehen, abzielen.

Windows Live Spaces[25] ist eine Blogseite mit sozialen Funktionalitäten. Das Netzwerk hat etwa 27 Mio Mitglieder.

Xanga[26] ist eine Blogseite mit etwa 40 Mio Mitgliedern. Der Großteil der Mitglieder sind Teenager.

[19] http://www.ning.com
[20] http://blog.ning.com/2008/02/185000_social_networks.html
[21] http://www.orkut.com
[22] http://www.plaxo.com
[23] http://www.skyrock.com/blog
[24] http://www.studivz.net
[25] http://home.services.spaces.live.com
[26] http://www.xanga.com

Xing[27] ist ein in Deutschland ansässiges berufsorientiertes soziales Online-Netzwerk. Es hat etwa 5 Mio Nutzer. Der Großteil der Nutzer kommt aus Deutschland. Xing hat eine mehrsprachige Benutzerschnittstelle, wo sogar Russisch oder Finnisch ausgewählt werden können. Es gibt die Möglichkeit eine Premium-Mitgliedschaft zu erwerben, die zusätzliche Funktionen freischaltet. Xing hat eine Jobbörse, wo die Nutzer Jobangebote erstellen oder sich für bereits vorhandene Jobangebote bewerben können.

2.2 Grundlagen

Im Folgenden werden die Grundlagen für diese Arbeit erläutert.

2.2.1 Semantic Web

Die Idee des Semantic Webs kam vom Tim Berners-Lee, dem Begründer des World Wide Webs. In seiner Vision sind Computer in der Lage, die gesamten Daten im Internet zu analysieren [LF99]. Diese Daten sind Inhalt, Links und Transaktionen zwischen Menschen und Computern. Das World Wide Web Consortium[28] (W3C) arbeitet daran, diese Vision Wirklichkeit werden zu lassen. Das erfordert die Erstellung von Standards und Technologien, um die Daten im Internet definieren und miteinander verknüpfen zu können, damit die Computer die Bedeutung der Daten erfassen und die Daten entsprechend weiterverarbeiten können.

2.2.2 Ressource Description Framework

Das Ressource Description Framework (RDF) [KC04] ist eine Sprache zur Darstellung von Metadaten. Sie ist ein Grundstein des Semantic Webs. Mit RDF werden zur Zeit Informationen über Webseiten bereitgestellt, die ein Internetsurfer beim Surfen auf der Webseite im Browser nicht angezeigt bekommt. Diese Informationen sind beispielsweise Titel, Name und Copyright. Ein RDF-Modell wird in XML [BPSM+04] oder Notation 3 (N3) [BL98] dargestellt. Es besteht aus Tripeln. Die Tripel haben die Form 'Subjekt Prädikat Objekt'. Das Subjekt ist die Ressource, die beschrieben wird. Das Prädikat ist die Eigenschaft. Das Objekt beschreibt den Wert des Prädikats. RDF eignet sich gut zur Beschreibung von Ontologien (siehe Abschnitt 2.2.3).

Ein einfacher Sachverhalt ist in der Abbildung 2 mit drei Tripeln dargestellt. 'PersonA' hat den Namen Peter und ist ein Elternteil von 'PersonB'.

[27] https://www.xing.com
[28] http://www.w3.org

'PersonB' hat den Namen Anna.

```
(PersonA, hatName, Peter)
(PersonA, istElternteilVon, PersonB)
(PersonB, hatName, Anna)
```

Abbildung 2: Beispiel für die Verwendung von Tripeln

2.2.3 Ontologie

Eine Ontologie ist in der Informatik eine explizite Spezifikation einer Konzeptualisierung [Gru93]. Sie hat folgende Bestandteile:

Klassen sind Beschreibungen gemeinsamer Eigenschaften.

Instanzen sind Objekte, die zu bestimmten Klassen gehören.

Relationen beschreiben die Beziehungen zwischen den Instanzen. Eine äquivalente Bezeichnung für Relation ist Eigenschaft.

Axiome sind Gesetzmäßigkeiten, die ohne Beweis wahr sind.

2.2.4 RDF-Schema

Das Resource Description Framework Schema (RDFS) [DB04b] ist eine W3C-Empfehlung. Mit RDFS werden Ontologien für bestimmte Domänen definiert. Die Ressourcen einer Domäne mit ihren Eigenschaften und Relationen können mit RDFS dargestellt werden. RDFS eignet sich zur Formalisierung einfacher Ontologien. Klassen und Eigenschaften werden einzeln modelliert. Das Klassenkonzept ermöglicht, eine formale Beschreibung der Semantik der verwendeten RDF-Elemente festzulegen.

Ein Beispielschema in N3 ist in der Abbbildung 3 dargestellt. Das Schema definiert die Klassen und Eigenschaften für den Sachverhalt aus der Abbildung 2. Auf die Kommentare im Schema wurde verzichtet, da die Namen selbsterklärend sind.

2.2.5 Friend-of-a-Friend

Friend-of-a-Friend (FoaF) [DB04a] ist eine der ersten Anwendungen von Semantic-Web-Technologien. Es ist eine Ontologie (siehe Abschnitt 2.2.3). FoaF erlaubt eine maschinenlesbare Beschreibung der personenbezogenen

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

Person a rdfs:Class
.

Mann a rdfs:Class
; rdfs:subClassOf Person
.

Frau a rdfs:Class
; rdfs:subClassOf Person
.

hatName a rdf:Property
; rdfs:domain Person
; rdfs:range rdfs:Literal
.

istElternteilVon a rdf:Property
; rdfs:domain Person
; rdfs:range Person
.
```

Abbildung 3: Ein Beispiel für die Verwendung von RDFS

Daten und der Beziehungen zwischen Personen. Die personenbezogenen Daten sind zum Beispiel der Name, der Wohnort, die Interessen, die eigenen Blogs oder die Bilder, auf denen man abgebildet ist. Das Vokabular, das für die Beschreibung verwendet werden kann, ist durch ein RDF-Schema definiert. Die aus dem Vokabular verwendeten Klassen und Eigenschaften bilden ein XML-basiertes RDF-Dokument.

Jede Person kann Informationen zusammentragen und als FoaF-Dateien veröffentlichen. Eine FoaF-Datei kann auf folgende Weisen erstellt werden:

- mit einem Generator, zum Beispiel FoaF-a-matic[29]
- aus einem sozialen Online-Netzwerk, das eine Möglichkeit bietet, FoaF-Dateien aus seinen Profildaten zu erzeugen, zum Beispiel LiveJournal.com

[29]http://www.ldodds.com/foaf/foaf-a-matic.de.html

- durch Konvertierung von Jabber-Konten[30] oder vCard[31]-Dateien mit einem Werkzeug, wie zum Beispiel FOAFgen[32]
- manuell, wenn man sich mit der FoaF-Spezifikation und RDF auskennt.

FoaF-Dateien können analysiert und ausgewertet werden oder mittels FoaF-Explorer[33] angeschaut werden. Neben der Analyse sind mehrere weitere Verwendungsmöglichkeiten für FoaF möglich. Diese sind zum Beispiel Anwendung in Office-Programmen als Datenbasis für Serientexte, in Job-Börsen oder als Transferformat der persönlichen Daten zwischen unterschiedlichen sozialen Online-Netzwerken.

Ein Beispiel ist in der Abbildung 4 dargestellt.

2.3 Verwandte Arbeiten

In diesem Abschnitt werden Projekte vorgestellt, die auch Daten aus sozialen Online-Netzwerken auslesen können.

2.3.1 Socialstream

Socialstream[34] ist das Ergebnis eines von Google gesponserten Projekts an der Carnegie Mellon Universität.

Das Projektziel war, über die Nutzung der sozialen Online-Netzwerke nachzudenken und gegebenenfalls neue Nutzungsmöglichkeiten zu entdecken. Der Schwerpunkt lag auf dem Entdecken der Benutzerbedürfnisse in den sozialen Online-Netzwerken und auf der Erforschung vereinheitlichter Netzwerke, wie sie die Benutzererfahrung erhöhen. Es wurde ein Versuch unternommen, ein System zu entwerfen, welches es ermöglichen sollte, soziale Inhalte über mehrere Netzwerke hinweg anzuschauen, zu beantworten und zu teilen.

Das Projekt war darauf ausgerichtet, bei der Verbesserung des sozialen Online-Netzwerks Orkut zu helfen. Es sollte nicht nur die Benutzerschnittstelle neu entworfen werden. Das Team überlegte, auf welche Weise die Benutzer der sozialen Online-Netzwerke durch die Nutzung einen Mehrwert bekommen können.

[30] http://dougal.gunters.org/jabfoaf

[31] vCard: 'elektronische Visitenkarte', kann mit vielen Email-Programmen erzeugt werden

[32] http://www.toxi.co.uk/foafgen

[33] http://xml.mfd-consult.dk/foaf/explorer

[34] http://hcii.cmu.edu/M-HCI/2006/SocialstreamProject/index.php

Es entstand ein System, das dem Nutzer nahtloses Teilen und Ansehen verschiedener Inhalte über mehrere Netzwerke hinweg ermöglicht. Das System aggregiert Daten und Verbindungen aus verschiedenen sozialen Online-Netzwerken. Diese Informationen werden unter einer Oberfläche dargestellt. Die Verwaltung der Informationen ist möglich.

Socialstream kann die im Abschnitt 1.1 genannten Probleme nicht lösen. Es ist zwar möglich alle sichtbaren Daten aus den Netzwerken zu sehen, jedoch fehlt die Möglichkeit, die Daten strukturiert und einheitlich zu speichern. Über die Erkennung gleicher Profile wurden keine Informationen gefunden. Aufgrunddessen ist anzunehmen, dass es keine automatische Erkennung gibt. Diese Erkennung ist zum Beispiel bei Problem 1 erforderlich, um seine Freunde in unterschiedlichen sozialen Online-Netzwerken automatisch zu finden.

2.3.2 Flock

Flock[35] ist ein freier Internetbrowser. Er basiert auf Mozilla Firefox. Von seinen Entwicklern wird Flock als 'Nextgeneration' Internetbrowser bezeichnet. Er wurde entwickelt, um den Menschen neue Möglichkeiten der Internetnutzung zu eröffnen. Die Menschen sollen sich am Internet aktiv beteiligen. Es wurden der Zugang und die Nutzung sozialer internet-basierter Anwendungen vereinfacht. Die zu der Vereinfachung benötigten Funktionen wurden direkt in den Flockbrowser integriert. Flock ermöglicht ohne Umstände das Teilen von Fotos, Videos, Blogs, Feeds und Comments innerhalb von sozialen Online-Netzwerken, Media Providern und populären Internetseiten.

Das bekannteste zurzeit unterstützte soziale Online-Netzwerk ist Facebook. Andere unterstützte Netzwerke sind Twitter Nanoblogging-Service, Bilder-Hosting-Portale Flickr und Photobucket, Video-Hosting-Portal Youtube, Blog-Hosting-Seiten Blogger, Blogsome, LiveJournal, Wordpress, Xanga und Typepad, Lesezeichen-Hosting-Seiten 'de.icio.us' und 'Ma.gnolia' und Piczo, das soziale Online-Netzwerk für Teenager.

Flock fördert zwar die Übersichtlichkeit beim Surfen durch die unterstützten Netzwerke, jedoch sind hier im Wesentlichen die gleichen Kritikpunkte anzubringen wie bei OpenSocial, bezüglich der im Abschnitt 1.1 beschriebenen Probleme.

2.4 Bestehende Schnittstellen

Im Folgenden werden bestehende Schnittstellen, die einige soziale Online-Netzwerke zum Datenaustausch anbieten, vorgestellt.

[35] http://www.flock.com

2.4.1 OpenSocial

OpenSocial ist eine Programmierschnittstelle für Anwendungen in sozialen Online-Netzwerken. Es wurde von Google entwickelt. Das Entwicklungsziel war, den Zugriff auf soziale Online-Netzwerke für Entwickler zu vereinfachen. Durch den Einsatz von OpenSocial brauchen die Entwickler nicht mehr ihre Anwendungen an die unterschiedlichen Schnittstellen verschiedener Netzwerke anzupassen. Man baut eine OpenSocial-Anwendung, die man in verschiedene Netzwerke integriert. Diese Netzwerke müssen allerdings OpenSocial unterstützen.

Die Hauptfunktionalitäten von OpenSocial umfasst den Zugriff auf die Kernfunktionen und auf die Informationen der sozialen Online-Netzwerke. OpenSocial besteht aus drei Teilen. Diese sind jeweils Zugriffe auf die Profildaten, auf die Beziehungen zu anderen Nutzern und auf die Aktivitäten der Nutzer.

Es haben bereits mehrere Netzwerke angekündigt, OpenSocial zu unterstützen. Die wichtigsten dabei sind: LinkedIn, Friendster, Ning, Plaxo, Salesforce und Googles Orkut. Auch Xing kündingte eine Zusammenarbeit an. Seit der Veröffentlichung von OpenSocial am 1.11.2007 wurde die Unterstützung von Plaxo und Ning implementiert.

2.4.2 Facebook-API

Facebook stellt dem Entwickler eine Schnittstelle zum Datenzugriff zur Verfügung. Diese Schnittstelle basiert auf REST, d. h. die Methoden der Schnittstelle werden über HTTP POST und GET aufgerufen. Die Schnittstelle gewährt Lesen, Einfügen und Ändern der Daten in authentifiziertem Facebookprofil.

Einige von der Schnittstelle angebotenen Methoden sind:

- Man kann eine gezielte Anfrage für zwei beliebige Nutzer-IDs abschikken. Das Ergebnis der Anfrage ist ein boolscher Wert. Der Wert sagt aus, ob diese zwei Nutzer Freunde sind oder nicht.
- Es ist möglich, eine Liste aller IDs der Freunde des aktuellen Nutzers zu erhalten.
- Eine Liste der Teilnehmer einer spezifizierten Gruppe kann abgefragt werden.
- Die ID des Nutzers, zu dem die aktuelle HTTP-Verbindung zugewiesen ist, kann angefordert werden.

- Alle Profildaten zu einer gegebenen ID, die im Browser für den Nutzer sichtbar sind, können erhalten werden.
- Eine FQL-Query wird ausgeführt. Das Ergebnis der Query wird zurückgegeben.

FQL ist eine SQL-ähnliche Sprache. Queries müssen formuliert und an Facebook-API verschickt werden, um verschiedene Daten abzufragen.

Beispiel: Mit der Anfrage `'SELECT name FROM user WHERE uid=123456 OR uid=654321'` erhält man die Namen zu den gegebenen IDs.

Die Vorteile der Nutzung von FQL sind:

- Mit FQL kann der Datenverkehr zwischen Client und Server reduziert werden. Die Queries können mehrere einschränkende Bedingungen enthalten, welche weitere Methodenaufrufe ersparen.
- Die Anzahl der Anfragen an den Server wird reduziert. Anstatt eine Methode aufzurufen und die Rückgabe in einem Aufruf einer anderen Methode zu verwenden, bastelt man eine FQL-Query und lässt diese ausführen.
- FQL bietet eine konsistente, einheitliche Schnittstelle für alle Daten. Die Entwickler brauchen sich nicht die vielen Methoden mit unterschiedlichen Rückgabetypen und unterschiedlichen Parametern zu merken, wenn sie FQL verwenden.

Einige Beispiele von FQL-Queries, die man in fql.query anstatt von anderen Facebook-API-Methoden aufrufen kann, um das gewünschte Ergebnis zu erhalten, sind in der Abbildung 5 dargestellt.

2.5 Duplikaterkennung

Im Folgenden werden die Grundlagen der in dieser Arbeit erstellten Erkennung der Profile von gleichen Personen aus unterschiedlichen Netzwerken erläutert.

2.5.1 Levenshtein-Distanz

Die Levenshtein-Distanz wurde von Wladimir Levenshtein eingeführt [Lev66]. Sie ist ein Maß für den Unterschied zwischen zwei Zeichenketten. Das Maß wird bestimmt, indem man die Anzahl der Operationen Einfügen, Löschen und Ersetzen berechnet, um die eine Zeichenkette in die andere zu überführen.

Mit der Berechnung der Levenshtein-Distanz wird die Ähnlichkeit von Zeichenketten bestimmt. Beispielhafte Einsatzgebiete sind Rechtschreibprüfung und Duplikaterkennung.

Zur Berechnung der Distanz mit einem einfachen Algorithmus wird eine $(n + 1) \times (m + 1)$-Matrix benötigt. n und m sind die Längen der zu vergleichenden Zeichenketten. Die Berechnung erfolgt durch die folgende Rekursionsgleichung:

$$D_{0,0} = 0; D_{i,j} = minimum \begin{cases} D_{i-1,j-1} & +0 \text{ (gleicher Buchstabe)} \\ D_{i-1,j-1} & +1 \text{ (Ersetzung)} \\ D_{i-1,j} & +1 \text{ (Einfügung)} \\ D_{i,j-1} & +1 \text{ (Löschung)} \end{cases} \quad (1)$$

Die einzelnen Schritte im Algorithmus für die Zeichenketten s und t sind:

1. weise n die Länge von s zu, weise m die Länge von t zu, falls n=0 m ausgeben, falls m=0 n ausgeben, konstruiere eine $(n + 1) \times (m + 1)$-Matrix

2. initialisiere die Zeile 0 von 0 bis n, initialisiere die Spalte 0 von 0 bis m

3. Schleife über alle Zeichen von s (i von 1 bis n), Schleife über alle Zeichen von t (j von 1 bis m), falls s[i]=t[j] dann Kosten 0 zuweisen, falls s[i]≠t[j] dann Kosten 1 zuweisen

4. weise d[i,j] das Minimum von $d[i-1, j] + 1$, $d[i, j-1] + 1$ und $d[i-1, j-1] + Kosten$ zu

5. nachdem alle Iterationen abgearbeitet sind, steht das Ergebnis in d[n,m]

2.5.2 Ontology Mapping

Unter 'Ontology Mapping' versteht man die Zusammenführung mehrerer Ontologien. Mark Ehrig beschreibt in einer Arbeit [ES] die Regeln und einen Algorithmus, um die Zusammenführung zu bewerkstelligen.

Die für diese Arbeit interessanten Regeln sind:

R1: Wenn die 'Labels'[36] gleich sind, dann sind wahrscheinlich die Entitäten gleich.

R13: Wenn zwei Instanzen mit der gleichen Eigenschaft zu einer anderen Instanz verbunden sind, dann sind die zwei Instanzen ähnlich.

[36]Label: Beschriftung

R15: Ontologie-Beschreibungssprachen, zum Beispiel 'Web Ontology Language' (OWL) [DS04], können Eigenschaften enthalten, die Gleichheit zwischen Klassen ('equivalentClass' in OWL), Eigenschaften ('equivalentProperty') und Instanzen ('sameAs' in OWL) explizit definieren. Wenn zwei Klassen, zwei Eigenschaften oder zwei Instanzen mit diesen Eigenschaften verbunden sind, dann sind sie gleich.

Für die Ähnlichkeit zwischen den Ressourcen in einer Ontologie gilt die Transitivität. A und B sind ähnlich. B und C sind ähnlich. Daraus folgt: A und C sind ähnlich.

Der Algorithmus von Mark Ehrig, der die Regeln zur Zusammenführung der Ontologien verwendet, sieht folgendermaßen aus:

Schritt 1: Man hat zwei Ontologien.

Schritt 2: Die unabhängige Ähnlichkeit der Entitäten zwischen den Ontologien wird berechnet. Es werden die Regeln R1, R2 und R15 angewendet.

Schritt 3: Alle Regeln werden unter Berücksichtigung der Ergebnisse vom vorhergehenden Schritt angewendet. Man kann auch eine Teilmenge der Regeln anwenden.

Schritt 4: Die Schritte 2 und 3 werden solange durchgeführt, bis es keine signifikanten Änderungen in der Menge der erkannten ähnlichen Ressourcen mehr gibt.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

<foaf:Person xml:lang="en">

   <!-- Name -->
    <foaf:name>Peter Meier</foaf:name>

    <!-- Email -->
    <foaf:mbox rdf:resource="mailto:peter@meier.de"/>

    <!-- Webseite -->
    <foaf:homepage rdf:resource="http://www.meier.de"/>

    <!-- Foto -->
    <foaf:depiction
      rdf:resource="http://www.meier.de/fotos/london/london034.jpg" />

    <!-- Die Person kennt -->
    <foaf:knows>

      <foaf:Person>
        <foaf:name>Hans Schmidt</foaf:name>
      </foaf:Person>

    </foaf:knows>

  </foaf:Person>

</rdf:RDF>
```

Abbildung 4: FoaF-Beispiel

```
Anfrage der IDs der Freunde zu einer gegebener ID:
  SELECT uid2 FROM friend WHERE uid1=loggedInUid.

Anfrage, ob Freunde oder nicht:
  SELECT uid1, uid2 FROM friend WHERE uid1=uid1 AND uid2=uid2

Query, um die Gruppenteilnehmer zu erhalten:
  SELECT uid, gid, positions FROM group_member WHERE gid=gid

Profildaten eines Nutzers:
  SELECT uid, fields FROM user WHERE uid IN (uids)
```

Abbildung 5: FQL-Beispiele

3 Entwurf

Dieses Kapitel beinhaltet den Kern dieser Arbeit. Es wird die Kernidee erläutert und die Anforderungen an das System erstellt. Drei Netzwerke werden ausgewählt. Diese werden auf die in ihnen sichtbaren Daten analysiert und es wird ein einheitliches Datenmodell entworfen. Es wird der Entwurf des Systems beschrieben. Die einzelnen Teile sind Datenextraktion, Datenaggregation, Datenzugriff, Erkennung redundanter Kontakte und der Entwurf der Architektur.

3.1 Kernidee

Das zentrale Thema dieser Arbeit besteht aus drei Teilen. Im ersten Teil wurden soziale Online-Netzwerke analysiert. Die Analyse umfasst, welche nutzerrelevanten Daten in sozialen Online-Netzwerken vorkommen. Bei diesen Daten handelt es sich primär um die Kontakte des Nutzers, die Kontakte dieser Kontakte, die persönlichen Informationen über den Nutzer und die persönlichen Informationen über die Kontakte des Nutzers. Als sekundär werden die persönlichen Daten und die Kontakte aller anderen Nutzer betrachtet, da hier mit Einschränkungen der Zugriffe auf diese Daten zu rechnen ist. Aufgrund dieser Analyse wurde ein Datenmodell entworfen. Das Datenmodell kann zum einen die unterschiedlichen nutzerrelevanten Daten aus sozialen Online-Netzwerken aufnehmen. Zum anderen ist es unkompliziert zu verarbeiten, um problemlosen Zugang für verschiedene Analysen auf diesen Daten zu ermöglichen.

Im einheitlichen Datenmodell werden die Daten in RDF (siehe Abschnitt 2.2.2) abgelegt. Es wurde ein RDF-Schema (siehe Abschnitt 2.2.4) aufgestellt. Eine RDF-Datenbank speichert die Daten aus den Netzwerken in diesem RDF-Schema.

Im zweiten Teil wurde ein System entworfen und implementiert. Dieses System kann die Profildaten aus sozialen Online-Netzwerken extrahieren. Diese Daten werden vom System in das im ersten Teil entworfene Datenmodell transformiert.

Abschließend wurde eine mögliche Analyse der gelesenen Daten vorgestellt und implementiert. Bei der Analyse handelt es sich um die Erkennung der Profile von gleichen Personen aus verschiedenen sozialen Online-Netzwerken.

3.2 Aufstellung der Systemanforderungen

Die zu entwerfende Software sollte in der Lage sein, die im Abschnitt 1.1 beschriebenen Probleme zu lösen. Es wurde eine Analyse vorgenommen, welche Funktionen die einzelnen Probleme zur Lösung benötigen.

Problem 1: Globale, netzwerkübergreifende Suche nach Personen. Zur Lösung des Problems muss man auf die Suchschnittstelle in den sozialen Online-Netzwerken zugreifen können. Die Ergebnisse müssen extrahiert werden. Bei Bedarf sollen die Ergebnisse mit bereits bekannten Freunden aus anderen Netzwerken verglichen und die Profile gleicher Personen aus unterschiedlichen Netzwerken erkannt werden.

Problem 2: Übersicht aller Kontakte mit persönlichen Daten aus unterschiedlichen Netzwerken. Man muss die Daten extrahieren können.

Problem 3: Übersicht der redundanten Kontakte aus verschiedenen Netzwerken. Man muss die Daten extrahieren und die Profile gleicher Personen erkennen können.

Problem 4: Lokales Speichern der eigenen Daten und der Daten der eigenen Kontakte. Man muss die Daten extrahieren und in strukturierter Form speichern können.

Problem 5: Berechnung des 'sozialen Kapitals' eines Profils. Man muss die Daten extrahieren können. Die Erkennung der Profile gleicher Personen aus verschiedenen Netzwerken sollte funktionieren, um das 'soziale Kapital' über mehrere Netzwerke hinweg zu berechnen.

Problem 6: Beantwortung der Fragestellungen von Netzwerkforschern. Man muss die Daten extrahieren, die Profile von gleichen Personen aus unterschiedlichen Netzwerken erkennen, die Daten strukturiert speichern und nach Personen in den Netzwerken suchen können.

Die Analyse hat folgende **Systemanforderungen** ergeben:

1. vollständiges Einlesen aller für den Nutzer sichtbaren Daten
2. Speichern der extrahierten Daten in einem einheitlichen Datenmodell
3. Suche nach Personen in verschiedenen Netzwerken
4. Erkennung der Profile von gleichen Personen aus unterschiedlichen Netzwerken.

3.3 Auswahl der Netzwerke

Die Auswahl der Netzwerke erfolgte nach folgenden Kriterien: Popularität, geografische Verbreitung und Zielgruppe. Die ausgewählten Netzwerke sollten möglichst populär sein. Die Netzwerke sollten untereinander sowohl Un-

terschiede als auch Gemeinsamkeiten aufweisen. Wenn zum Beispiel zwei Netzwerke ähnliche geografische Verteilung aufweisen, sollten ihre Zielgruppen verschieden sein, um ausgewählt zu werden. Die Auswahl erfolgte aus den im Abschnitt 2.1 beschriebenen Netzwerken.

Wegen großer Beliebtheit, vor allem im englischsprachigen Raum, wurde Facebook ausgewählt. Viele Teilnehmer in Facebook sind Studenten.

Das zweite ausgewählte Netzwerk ist StudiVZ. Es ist das beliebteste deutschsprachige soziale Online-Netzwerk. Die Teilnehmer sind zum größten Teil Studenten aus dem deutschsprachigen Raum. Durch diese Wahl wird es ermöglicht, die Daten englischsprachiger Studenten mit den Daten deutschsprachiger vergleichen zu können.

Das dritte ausgewählte Netzwerk ist das beruflich orientierte soziale Online-Netzwerk Xing. Laut Alexa.com gibt es keine gefragtere Seite in Deutschland, die den gleichen Service anbietet. Die meisten Mitglieder kommen zwar wie bei StudiVZ aus Deutschland, jedoch hat Xing eine andere Zielgruppe. Der Großteil der Mitglieder hat ihre Ausbildung schon abgeschlossen.

3.4 Datenanalyse der ausgewählten Netzwerke

Der Entwurf des einheitlichen Datenmodells (siehe Abschnitt 3.2 Anforderung 2) erfordert die Durchführung einer Datenanalyse der ausgewählten Netzwerke. In diesem Unterkapitel werden die Anforderungen, um die Datenanalyse durchführen zu können, und die Ergebnisse der Datenanalyse vorgestellt. Um die Präsentation der Ergebnisse übersichtlich zu halten, wurden die Daten in Gruppen zusammengefasst.

In den meisten sozialen Online-Netzwerken können nur Mitglieder navigieren. Die Mitglieder müssen dabei eingeloggt sein. Um eine Datenanalyse in einem Netzwerk durchzuführen, ist es deshalb erforderlich, mindestens ein Benutzerkonto in diesem Netzwerk anzulegen. Dieses Benutzerkonto sollte natürlich auch mindestens einen Freund bekommen. Damit wird sichergestellt, dass man auch die Beziehungen analysieren kann. Es ist sinnvoll, ein zweites Benutzerkonto im zu analysierenden Netzwerk anzulegen und dieses mit dem ersten freundschaftlich innerhalb des Netzwerkes zu verknüpfen. Der Vorteil des zweiten Kontos ist, dass man die Änderungen und Einstellungen von dem zweiten Benutzerkonto aus jederzeit überprüfen und testen kann. Nicht nur persönliche Daten können auf diese Weise gleich überprüft werden, sondern auch die unterschiedlichen Sicherheitseinstellungen.

Die Angabe der persönlichen Informationen geschieht in allen Netzwerken auf eine von zwei Arten: Einige Informationen, wie zum Beispiel die Interessen, werden als Freitext eingegeben. Andere Angaben werden durch eine Auswahl aus vorgegebenen Begriffen gemacht. Das schliesst das Vertippen

des Nutzers beim Eingeben der Informationen aus. Beispiele hierfür sind die Angabe des Geburtsdatums und des Geschlechts.

3.4.1 Allgemeine persönliche Informationen

Der Name wird in jedem untersuchten Netzwerk bei der Erstellung eines Nutzerkontos angegeben. Dieser kann nicht nachträglich in den Einstellungen geändert werden.

Die Nutzer von Facebook haben die meisten Möglichkeiten, Angaben ihrer persönlichen Informationen zu machen. Das Geschlecht kann durch die Auswahl aus 'männlich' oder 'weiblich' angegeben werden. Der Geburtstag wird durch die Auswahl des Datums, des Monats und des Jahres angegeben. Folgende Angaben werden als Freitext gemacht: Heimatstadt, politische Gesinnung, Religion, Interessen, Aktivitäten, Lieblingsmusik, Lieblings-TV-Shows, Lieblingsfilme, Lieblingsbücher, Lieblingszitate und ein Text mit ein paar Sätzen über sich selbst. Angabe von früheren Namen ist auch möglich. Bei der Angabe des Beziehungsstatus muss aus folgenden sechs Möglichkeiten eine gewählt werden: 'Single', 'in einer Beziehung', 'verlobt', 'verheiratet', 'es ist kompliziert' oder 'offene Beziehung'. Bei allen Auswahlmöglichkeiten ausser 'Single' kann der Name der Person, mit der man diese Beziehung hat, angegeben werden. Befindet sich diese Person ebenfalls in Facebook und ist die Person in der eigenen Freundesliste, wird auf diese Person verlinkt. Auch gibt es die Möglichkeit der Angabe, an welchem Geschlecht man interessiert ist. Hier kann 'männlich', 'weiblich' oder 'beides' angegeben werden. Schließlich gibt es noch die Angabe, was man im Netzwerk sucht. Hierzu können die Begriffe 'Freundschaft', 'Dating', 'Beziehung' und 'Networking' beliebig kombiniert werden.

StudiVZ bietet als Facebook-Kopie in etwa die gleichen Möglichkeiten, persönliche Angaben zu machen. Folgende Unterschiede zu Facebook konnten ausgemacht werden: Das Textfeld Aktivitäten heisst hier 'Clubs/Vereine', was im Grunde genommen das Gleiche ist, da eine Teilnahme im Verein eine Aktivität darstellt. Die Information 'Heimatland' hat ein eigenes Textfeld. Diese Information steht in Facebook unter 'Heimatstadt', wenn Facebook die Stadt erkannt hat. Bei der Angabe des Beziehungsstatus gibt es die Auswahlmöglichkeit 'Romanze' statt 'verlobt' bei Facebook. Die Person, zu der man die Beziehung hat, kann nicht angegeben werden. 'Politische Gesinnung' ist kein Freitext, sondern Auswahl einer aus einer vorgegebenen Menge. Die Angabe, was man im Netzwerk sucht, ist eine Kombination aus sechs verschiedenen Begriffen. Diese sind 'nette Leute', 'Lern-/Übungsgruppen', 'Sprachpartner', 'Dating', 'Parties' und 'was sich eben ergibt'. Die Angaben der Lieblings-TV-Shows, der Religion und an welchem Geschlecht man inte-

ressiert ist, sind nicht möglich.

Die möglichen Angaben der persönlichen Informationen in Xing unterscheiden sich stark von denen in Facebook und StudiVZ. Die Angaben in Xing haben zum größten Teil etwas mit dem Beruf zu tun. Zusätzlich zum Namen können akademischer Titel und akademischer Abschluss angegeben werden. Die Angabe des Geburtstags ist auch möglich. Weitere Angaben sind Freitextfelder zu dem, was man sucht, was man bietet, welche Interessen man hat, mit welchen Organisationen man in Verbindung steht und warum man Xing-Nutzer ist. Die Angabe, warum man Xing-Nutzer ist, ist eine Kombination aus den Begriffen 'Neugeschäft und Aufträge generieren', 'neue Mitarbeiter finden', 'interessante Personen kennenlernen', 'mein Netzwerk pflegen', 'an Events teilnehmen', 'alte Bekannte und Kollegen wiederfinden' und 'neuen Job finden'.

Die Abbildung 6 enthält eine tabellarische Übersicht über die möglichen persönlichen Informationen der untersuchten Netzwerke. Das 'x' heisst, diese Angabe ist in diesem Netzwerk möglich.

3.4.2 Kontaktinformationen

Die Angaben zur Kontaktaufnahme sind in Facebook Email, Nicknamen in Instant Messagings, Mobiltelefonnummer, Festnetztelefonnummer, Anschrift und Webseite. Nicknamen können nur zu folgenden Instant Messagings angegeben werden: Skype, ICQ, AIM, Google Talk, Windows Live, Yahoo, Gadu-Gadu.

In StudiVZ ist die Angabe der Email als Information für andere Nutzer nicht möglich. Als Angaben der Instant Messaging Nicknamen sind folgende möglich: ICQ, AIM, Jabber und Skype. Die anderen Angaben sind wie in Facebook möglich. Zusätzlich können das eigene Wohnheim und die Zimmernummer angegeben werden.

Der wichtigste Unterschied von Kontaktinformationen in Xing liegt in der Trennung der meisten Kontaktinformationen in privat und geschäftlich. So ist es in Xing möglich, Anschrift, Fesnetztelefon, Mobiltelefon, Fax und Email sowohl geschäflich als auch privat anzugeben. Die Angabe der Nicknamen ist in folgenden Instant Messagings möglich: Skype, ICQ, MSN, Yahoo, AIM, Jabber und Google Talk. Die eigene Webseite wird in Xing unter 'andere Profile' eingetragen. Hierzu zählen unter anderen auch eigene Blogs und eigene Profile. Beispiele für die Profile sind Youtube, Flickr, Second Life, Ebay und Amazon.

Die Abbildung 7 enthält eine tabellarische Übersicht über die möglichen Kontaktinformationen der untersuchten Netzwerke. Das 'x' heisst, diese Angabe ist in diesem Netzwerk möglich.

Information	Facebook	StudiVZ	Xing
Name	x	x	x
Geburtstag	x	x	x
Geschlecht	x	x	
Beziehungsstatus	x	x	
Beziehungspartner	x		
Religion	x		
Politische Gesinnung	x	x	
Aktivitäten/Clubs/Vereine	x	x	
Interessen	x	x	x
Grund, warum im Netzwerk	x	x	x
An welchem Geschlecht interessiert	x		
Heimatstadt	x	x	
Heimatland	x		
Lieblingsmusik	x	x	
Lieblings-TV-Shows	x		
Lieblingsbücher	x	x	
Lieblingsfilme	x	x	
Lieblingszitat	x	x	
Über mich	x	x	
Was sucht man			x
Was bietet man			x
Organisationen			x
Akademischer Titel			x
Akademischer Abschluss			x

Abbildung 6: Allgemeine persönliche Informationen

3.4.3 Angaben zur Bildung

Bei den Angaben zur Bildung können in Facebook mehrere besuchte Schulen und Hochschulen angegeben werden. Bei beiden ist die Angabe eines Jahres notwendig. Bei den Hochschulen können noch Angaben der Studienrichtungen, maximal drei, und die Art des Studiums, 'normal' oder 'Aufbaustudium', gemacht werden.

In StudiVZ kann man nur eine Hochschule angeben. Zusätzliche Angaben sind Studiengang mit maximal drei Studienrichtungen, seit wann man an dieser Hochschule ist und sein Status an dieser Hochschule. Der Status ist eine Auswahl aus 'Student', 'Hochschulmitarbeiter', 'Alumna/nus' oder

Information	Facebook	StudiVZ	Xing
Instant Messaging Nicknamen	x	x	x
private Email	x	x	x
geschäftliche Email			x
private Festnetztelefonnummer	x	x	x
geschäftliche Festnetztelefonnummer			x
private Mobiltelefonnummer	x	x	x
geschäftliche Mobiltelefonnummer			x
private Anschrift	x	x	x
geschäftliche Anschrift			x
private Faxnummer			x
geschäftliche Faxnummer			x
Webseite	x	x	x
andere Profile			x

Abbildung 7: Kontaktinformationen

'Abiturient/Maturand'.

Wie in Facebook können auch in Xing mehrere besuchte Hochschulen angegeben werden. Die zusätzlichen Angaben sind das Jahr und der Monat jeweils für Start und Ende des Besuchs, die Fachrichtung, der Abschluss und die Schwerpunkte des Studiums. Außerdem können mehrere Qualifikationen angegeben werden. 'System-Administrator (MCSA, MCSE)' ist zum Beispiel eine Qualifikation. Auch ist die Angabe der gesprochenen Sprachen möglich. Zu jeder Sprache kann optional eine Angabe bezüglich des Umfangs der Sprachkenntnisse gemacht werden. Diese Angabe muss 'Grundkenntnisse', 'Gut', 'Fließend' oder 'Muttersprache' sein.

Die Abbildung 8 enthält eine tabellarische Übersicht über die möglichen Angaben zur Bildung der untersuchten Netzwerke. Das 'x' heisst, diese Angabe ist in diesem Netzwerk möglich.

3.4.4 Angaben zur Arbeit

In Facebook können Angaben zu mehreren Arbeitstellen mit folgenden näheren Informationen gemacht werden: der Arbeitgeber, die eigene Position, die Lage, die Beschreibung der Arbeit, der Anfang und das Ende des Arbeitsverhältnisses.

In StudiVZ kann nur eine Arbeitstelle explizit angegeben werden. Wie in Facebook sind Angaben des Arbeitgebers, der Position und der Beschreibung möglich. Es existiert eine Auswahl mit der Bezeichnung 'Art des Jobs'. Hier

Information	Facebook	StudiVZ	Xing
Hochschule	x	x	x
weitere Hochschulen	x		x
Jahr an der Hochschule	x		
Art des Studiums	x		
An der Hochschule seit		x	x
An der Hochschule bis			x
Hochschulstatus		x	
Studiengang mit Fachrichtungen	x	x	x
Schwerpunkte des Studiums			x
Qualifikationen			x
Sprachen			x
Schulen	x		

Abbildung 8: Angaben zur Bildung

kann man einen Begriff aus einer Menge auswählen. Beispiele hierfür sind 'Praktikant' oder 'Papierschieber'. Außerdem gibt es ein Freitextfeld zum Eintragen seiner bisherigen Karriere.

Wie in Facebook können auch in Xing Angaben zu mehreren Arbeitstellen gemacht werden. Allerdings können diese Angaben erheblich detaillierter ausfallen. Zu jedem Arbeitgeber sind nicht nur Angaben seiner Position, der Beschreibung der Arbeit, des Anfangs und des Endes des Arbeitsverhältnisses, sondern auch der Industriebranche, des Status ('Teilzeit', 'Vollzeit', 'Praktikum' usw.), des eigenen Karrierelevels ('Berufseinsteiger', 'Geschäftsführer' usw.), der Art der Organisation ('gemeinnützig', 'selbständig' usw.), der ungefähren Firmengröße, der Homepage der Organisation und weiterer Branchen möglich. In Xing gibt es berufliche Angaben, die mit keinem Arbeitgeber im Zusammenhang stehen. Zum einen können Auszeichnungen mit Jahr und URL eingetragen werden. Zum anderen ist es möglich seinen aktuellen Status auf dem Arbeitsmarkt ('arbeitssuchend', 'im Ruhestand', 'Personalvermittler' usw.) anzugeben.

Die Abbildung 9 enthält eine tabellarische Übersicht über die möglichen beruflichen Angaben der untersuchten Netzwerke. Das 'x' heisst, diese Angabe ist in diesem Netzwerk möglich.

3.4.5 Bilder

Facebook bietet die Möglichkeit, Bilder hochzuladen. Man kann für sein Profil ein eigenes Bild festlegen, welches dann auf der Profilseite angezeigt wird.

Information	Facebook	StudiVZ	Xing
Eine aktuelle Arbeitstelle	x	x	x
Weitere aktuelle/ehemalige Arbeitstellen	x		x
Anfang und Ende des Arbeitsverhältnisses	x		x
Position	x	x	x
Beschreibung	x	x	x
Lage	x		
Art des Jobs		x	
Industrie			x
Weitere Branchen			x
Arbeitsstatus			x
Karrierelevel			x
Unternehmensgröße			x
Homepage			x
Art der Organisation			x
Auszeichnungen			x
Eigener Status auf dem Arbeitmarkt			x

Abbildung 9: Angaben zur Arbeit

Es können Alben angelegt werden. Ein Album hat einen Namen und kann eine Beschreibung und eine Ortsangabe haben. Den Alben können Bilder zugewiesen werden. Die Bilder können eine Überschrift und Angaben, welche Personen auf dem Bild sind, haben. Zu jedem Bild können Kommentare hinterlassen werden. Ein Kommentar hat einen Autor, einen Text und einen Erstellungszeitpunkt.

StudiVZ hat die gleichen Möglichkeiten wie Facebook. Xing bietet nur die Möglichkeit, ein Bild für sein Profil hochzuladen.

Die Abbildung 10 enthält eine tabellarische Übersicht über die möglichen Angaben zu Bildern der untersuchten Netzwerke. Das 'x' heisst, diese Angabe ist in diesem Netzwerk möglich.

Information	Facebook	StudiVZ	Xing
Eigenes Profilbild	x	x	x
Alben mit Fotos	x	x	
Kommentare	x	x	

Abbildung 10: Bilder

3.4.6 Besondere Informationen

In diesem Abschnitt werden die besonderen Informationen in den untersuchten Netzwerken erläutert. Eine gemeinsame Besonderheit von allen drei Netzwerken ist, dass jeder Nutzer ein Gästebuch aktivieren kann.

Facebook hat ein Konzept der Networks. Ein Nutzer kann mehreren Networks angeschlossen sein. Jedoch ist es nicht möglich, sich verschiedenen Networks mit dem gleichen Typ anzuschließen. Die Typen sind geographisch, beruflich und bildungsbezogen. Beispiele für geographisch sind Deutschland oder USA, für beruflich FZI oder Siemens, für bildungsbezogen Uni Karlsruhe oder MIT.

StudiVZ hat ein abgeschwächtes Network-Konzept. Hier sind die Networks die jeweiligen Hochschuleinrichtungen. Man ist mit der Angabe seiner Hochschule automatisch Mitglied im entsprechenden Network. Man kann nur einem Network angeschlossen sein. Eine weitere Besonderheit in StudiVZ ist die Angabe der eigenen Lehrveranstaltungen.

In Xing ist es möglich, persönliche Tags und Notizen an andere Nutzer zu heften.

Die Abbildung 11 enthält eine tabellarische Übersicht über die möglichen besonderen Angaben der untersuchten Netzwerke. Das 'x' heisst, diese Angabe ist in diesem Netzwerk möglich.

Information	Facebook	StudiVZ	Xing
Gästebuch	x	x	x
Networks	x	x	
Lehrveranstaltungen		x	
Persönlicher Tag			x
Notiz			x

Abbildung 11: Besondere Informationen

3.4.7 Gruppen

Ein wichtiges Konzept in sozialen Online-Netzwerken stellen Gruppen dar. Unter einer Gruppe versteht man ein Zusammenschluss von Netzwerkteilnehmern, um sich zu einem Thema auszutauschen. Der Austausch findet in einem Forum der Gruppe statt. Dieser Forum kann von den Gruppenteilnehmern auf die gleiche Weise genutzt werden wie die unzähligen Internetforen im Web. Es gibt Threads mit Antworten der Teilnehmer.

Eine Gruppe in Facebook hat einen Namen und eine Kategorie und kann eine Beschreibung, eine Webseite, aktuelle Informationen, ein Büro, eine

Email, eine Adresse, ein eigenes Profilbild, ein Gästebuch, mehrere Alben mit Bildern und ein Forum haben. Außerdem können die Gruppen Networks angehören.

Die StudiVZ-Gruppen unterscheiden sich von den in Facebook folgendermaßen. Es gibt keine Emailangabe, kein Gästebuch und keine Bilderalben.

Die Xing-Gruppen haben außer Name und Beschreibung keine weiteren Detailangaben. Aber jede Gruppe in Xing kann mehrere Foren haben.

Die Abbildung 12 enthält eine tabellarische Übersicht über die möglichen Angaben zu Gruppen der untersuchten Netzwerke. Das 'x' heisst, diese Angabe ist in diesem Netzwerk möglich.

Information	Facebook	StudiVZ	Xing
Gruppen	x	x	x
Name	x	x	x
Profilbild	x	x	
Kategorie	x	x	
Beschreibung	x	x	x
Webseite	x	x	
Aktuelle Informationen	x	x	
Büro	x	x	
Email	x		
Adresse	x		
Bilderalben	x		
Gästebuch	x		
Forum	x	x	x
mehrere Foren			x
Networkzugehörigkeit	x		

Abbildung 12: Gruppen

3.5 SNEQ-Datenmodell

Aufgrund der im Abschnitt 3.4 vorgestellten Daten, wurde zu jedem untersuchten Netzwerk ein RDF-Schema erstellt. Aus diesen RDF-Schemas wurde ein einheitliches Datenmodell, das SNEQ-Datenmodell, erstellt. Das SNEQ-Schema ist vollständig im Anhang A in N3 zu finden. Das entworfene SNEQ-Schema kann alle Profildaten aus den drei ausgewählten Netzwerken (Facebook, StudiVZ, Xing) aufnehmen.

Alle Ressourcen, Eigenschaften und Klassen im SNEQ-Schema haben eine Eigenschaft 'usedByNetwork'. Die Eigenschaft kann auf die Werte 'studivz',

'facebook' und 'xing' verweisen. Dadurch kann man leicht nur Ressourcen aus bestimmten Netzwerken aus einem RDF-Modell laden.

Es folgen eine Reihe von Abbildungen mit Erklärungen. Auf den Abbildungen sind alle Daten und Eigenschaften aus dem SNEQ-Schema dargestellt. Rechtecke stehen für Klassen. Pfeile bedeuten Eigenschaften zu anderen Klassen. Eigenschaften, die Text-Metadaten oder Zeitangaben enthalten, sind unter den Klassen aufgelistet. Es werden nicht alle Eigenschaften erklärt, da viele Namen der Eigenschaften selbsterklärend sind.

Abbildung 13: Die Klassen Person und Gruppe (Group) haben die Klasse Akteur (Actor) als Oberklasse. Das wird durch die Eigenschaft 'subClassOf' ausgedrückt. Unterklassen können alle Eigenschaften haben, die ihre Oberklasse haben kann. Eine Person kann in mehreren Gruppen Teilnehmer sein (hasGroup). Eine Gruppe kann mehrere Foren (DiscussionBoard) haben. Ein DiscussionBoard kann mehrere Themen (DiscussionTopic) haben (hasDiscussionTopic). Ein Thema kann mehrere Textbeiträge (Comment) haben (hasTopicComment). Ein Textbeitrag kann einen Autor haben. Eine Person kann mehrere Fotoalben (PhotoAlbum) haben. Ein Fotoalbum kann mehrere Fotos enthalten. Auf einem Foto können mehrere Personen abgebildet sein (contains). Ein Foto kann mehrere Textbeiträge haben.

Abbildung 14: Hier sind die Eigenschaften der Klasse Akteur dargestellt. Ein Akteur kann eine Beschreibung haben (hasDescription). Ein Akteur kann mehrere Fotoalben, ein Gästebuch (Wall), eine Webseite (Website) und ein Textfenster (MiniFeed) haben. Ein Akteur kann Networks angehören. Ein Fotoalbum kann einen Namen (hasPhotoAlbumName), eine Ortsangabe (hasPALocation) und eine Beschreibung (hasPADescription) haben. Ein Gästebuch kann mehrere Textbeiträge enthalten. Ein Textfenster kann mehrere besondere Textbeiträge (MiniFeedStories) haben. Ein besonderer Textbeitrag kann Information enthalten, um welche Art des Beitrags es sich handelt (hasMiniFeedStory).

Abbildung 15: Eine Person kann persönliche Notizen und Erinnerungsnotizen in Form von Textbeiträgen haben. Eine Person kann mehrere Personen als Freund, mehrere soziale Online-Netzwerke und mehrere Webprofile in Form von Webseiten haben. Fotos sind Unterklassen von Zeitobjekten (TimedObject). Zeitobjekte können einen Entstehungszeitpunkt haben. Besondere und allgemeine Textbeiträge sind Unterklassen von Textobjekten. Textobjekte sind Unterklassen von Zeitobjekten und können einen Text haben.

Abbildung 16: Diese Abbildung zeigt zum größten Teil einfache Eigenschaften von der Klasse Person. Eine Person kann Gründe der Netzwerkteilnahme (hasLookingFor), Beziehungsstatus (hasRelationshipStatus), Geburtstag (hasBirthday), Zeitpunkt des Netzwerkbeitritts (hasJoinDate),

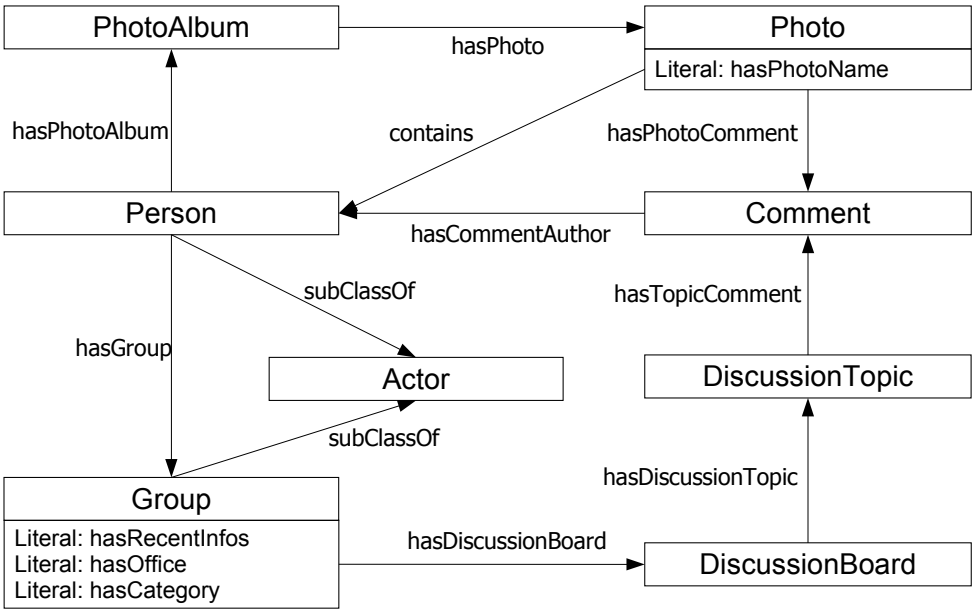

Abbildung 13: Akteur, Person, Gruppe und andere Klassen

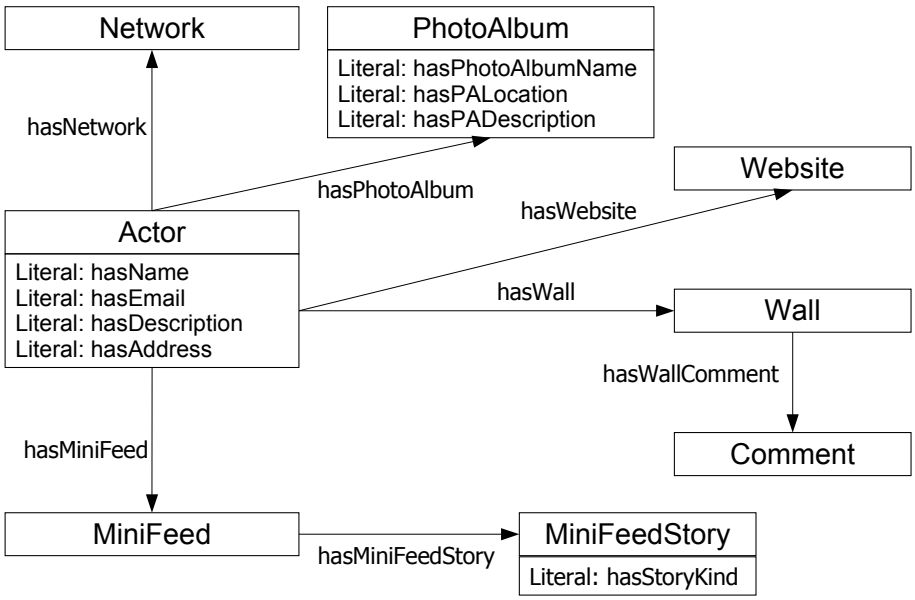

Abbildung 14: Eigenschaften von Akteur

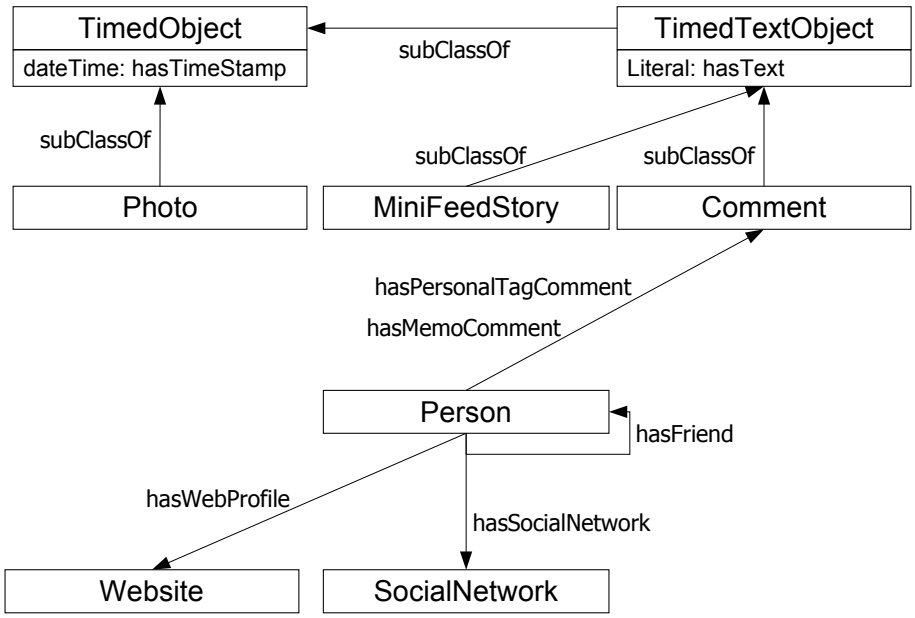

Abbildung 15: weitere Eigenschaften zwischen den Klassen

Zeitpunkt der letzte Profilaktualisierung (hasLastUpdate), Religion (hasReligiousView), politische Ansichten (hasPoliticalView), Lieblingsfilme (hasFavMovie), Lieblingszitate (hasFavQuote), berufliche Erfahrung (hasExperience), Auszeichnungen (hasAwards) und Arbeitsmarktstatus (hasStatus) haben. Die Klasse 'Sex' steht für Geschlecht. Eine Person kann von einem Geschlecht sein und an Personen von einem oder beiden Geschlechtern interessiert sein. Die Klasse 'Lection' steht für Vorlesung. Eine Person kann mehrere Vorlesungen haben.

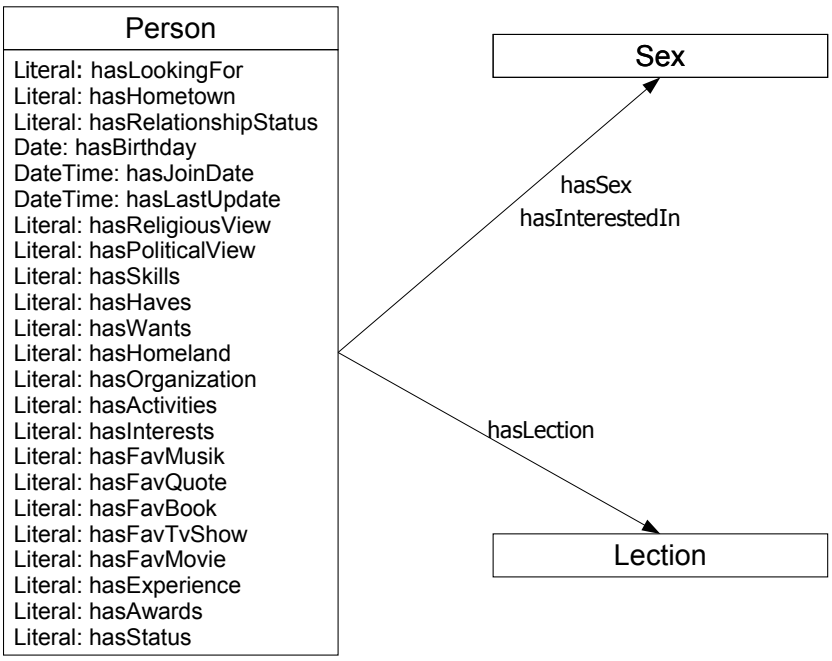

Abbildung 16: einfache Eigenschaften von Person

Abbildung 17: Diese Abbildung zeigt Beziehungen der Klasse Person zu anderen Klassen. Eine Person kann mehrere Arbeitstellen (Job) haben. Die Arbeitstelle kann den Arbeitgeber (hasEmployer), eine Position, eine Beschreibung (hasJobDescription) und eine Arbeitslage (hasWorkLocation) haben. Eine Person kann mehrere Bildungen (Education) haben. Eine Bildung kann einen Namen, eine Schule (hasHighScool), eine Hochschule (hasCollege), eine Sprache (hasLanguage), und einen Anfangs- und Endzeitpunkt haben. Eine Person kann mehrere 'Instant-Messaging'-Namen haben. Ein 'Instant-

Messaging'-Name hat eine Information mit dem eigentlichen Nicknamen und die Eigenschaft zu einem 'Instant Messaging'-Service zu gehören.

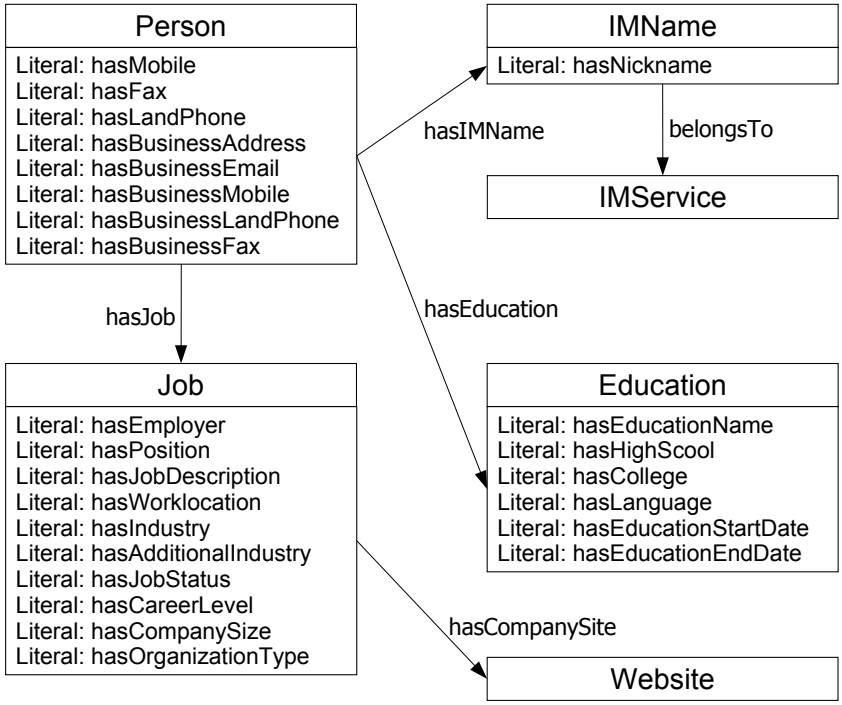

Abbildung 17: Eigenschaften von Person zu anderen Klassen

3.6 Datenextraktion

Im Folgenden wird beschrieben, wie die Datenextraktion aus den ausgewählten sozialen Online-Netzwerken stattfindet.

Nach der Datenanalyse und dem durchgeführten Entwurf des SNEQ-Schemas stellte sich die Frage, wie man an die Daten herankommen will. Hier gibt es zwei Möglichkeiten. Die erste: Man verwendet die Schnittstellen, die die Netzwerke anbieten. Die zweite: Man simuliert den HTTP-Verkehr eines Browsers, loggt sich in die Netzwerke ein und extrahiert die benötigten Daten aus dem vom Server zurückgelieferten HTML.

Trotz der Ankündigung von Xing, OpenSocial (siehe Abschnitt 2.4.1) zu unterstützen, sind die Schnittstellen noch nicht verfügbar (Stand: März 2008). Da weder Facebook noch StudiVZ OpenSocial unterstützen, fällt die Verwendung von OpenSocial aus. StudiVZ hat gar keine Schnittstelle. Die

Verwendung der in Abschnitt 2.4.2 vorgestellten Facebook-API hat zwar einige Vorteile, jedoch auch einige Nachteile. Nur durch die Facebook-API kommt man nicht an alle sichtbaren Daten in Facebook. So ist es zum Beispiel nicht möglich, die Diskussionsthemen und die Beiträge dazu über die API zu lesen. Hier ist man darauf angewiesen, auf die andere Art an diese Daten zu kommen. Außerdem ist das angeeignete Wissen aus der Verwendung der Facebook-API nicht anwendbar bei der Datenextraktion aus anderen Netzwerken. Aus den vorhergehenden Gründen wurde entschieden, die Daten über HTTP einzulesen. Das System simuliert das Surfverhalten der Netzwerknutzer. Die HTML-Seiten werden in XHTML [ea02] transformiert. Dadurch erleichtert man sich die Suche nach den richtigen Elementen, in welchen die benötigten Informationen stehen. Der Zugriff auf die Elemente wird mit XPATH-Anfragen [CD99] durchgeführt. Arbeiten, in denen XML aus HTML extrahiert wurde, sind zum Beispiel [Kur06] und [Völ03].

3.6.1 HTTP-Zugriffskomponente

Die von dem Softwaresystem an die HTTP-Zugriffskomponente gestellten Anforderungen sind:

- Mit der Komponente soll man sich in die sozialen Online-Netzwerke einloggen können.
- Mit der Komponente soll man innerhalb der sozialen Online-Netzwerke navigieren können.
- Die Komponente soll 'Cookies'[37] unterstützen und verschiedene Möglichkeiten des Cookiemanagements haben, falls die sozialen Online-Netzwerke auf verschiedene Weisen 'Cookies' setzen und handhaben.

Bei der Wahl der HTTP-Zugriffskomponente wurde entschieden, den HTTP-Client von Apache[38] zu verwenden. Alle Anforderungen werden von dem Apache-HTTP-Client zufriedenstellend erfüllt.

Der Apache-HTTP-Client war lange Zeit ein Bestandteil des Jakarta-Projekts. Er wurde 2007 zu einem eigenständigen Projekt hochgestuft. Diese Wahl erfolgte vorwiegend aufgrund mangelnder Alternativen von freien HTTP-Zugriffskomponenten und aufgrund der Tatsache, dass dieses Projekt gut gepflegt wird.

[37] http://de.wikipedia.org/wiki/Cookie
[38] http://hc.apache.org/httpclient-3.x

3.7 Datenaggregation und -zugriff

In diesem Abschnitt wird erklärt, wie die aus den Netzwerken extrahierten Daten im einheitlichen Format abgelegt werden und wie man auf diese Daten zugreift.

Die **Aggregation** der extrahierten Daten erfolgt mit Hilfe eines RDF-Modells. Das RDF-Modell wird mit dem SNEQ-Schema initialisiert. Das RDF-Modell kann die enthaltenen Daten jederzeit im lokalen Speicher ablegen. Die Daten können vor dem Speichern auch direkt verwendet werden. Außerdem können die Daten aus dem lokalen Speicher zu einem beliebigen späteren Zeitpunkt in das RDF-Modell geladen werden. Das kann zum Beispiel geschehen, um festzustellen, ob man bereits bestimmte Daten zu einem Profil extrahiert hat. Die Daten können für die Untersuchung der Profile von gleichen Personen aus verschiedenen Netzwerken verwendet werden. Auch kann eine Ausgabe auf dem Bildschirm in RDF-Syntax, zum Beispiel in der 'Turtle-Syntax' [BBL08], erzeugt werden.

Der **Zugriff** auf die Daten erfolgt mit RDFReactor [Völ06]. Dieses Werkzeug wurde am FZI[39] entwickelt. Es ermöglicht objektorientierten Zugriff in Java. Die Kenntnisse über RDF-Schema sind für die Benutzung des RDFReactor ausreichend. Die Java-Interfaces werden automatisch aus dem SNEQ-Schema generiert. Auf diese Weise vereinfacht RDFReactor die Verwendung von RDF für Java-Entwickler. RDFReactor verwendet die Schnittstellen von RDF2Go[40], welches ebenfalls am FZI entwickelt wurde. RDF2Go ist ein Werkzeug, welches über den RDF-Datenspeicher abstrahiert.

Der Zugriff kann sowohl über die RDF-Datenspeicher-Methoden als auch mit SPARQL [PS08] erfolgen. SPARQL ist eine Abfragesprache für RDF. Sie wurde vom W3C am 15.01.08 endgültig als Rekommendation freigegeben. Die Anfragen haben eine SQL-ähnliche Syntax.

3.8 Erkennung der Profile gleicher Personen

Die im Abschnitt 2.5.2 vorgestellte Vorgehensweise zum Zusammenführen der Ontologien wurde analysiert und auf die Bedürfnisse der Erkennung der Instanzengleichheit im entworfenen SNEQ-Datenmodell (siehe Abschnitt 3.5) angepasst.

Im SNEQ-Datenmodell ist die Gleichheit der Eigenschaften und Klassen gegeben. Die zu vergleichenden Profile kommen zwar aus unterschiedlichen Netzwerken, sind aber Instanzen der gleichen Klasse (sneq:Person) und

[39] http://www.fzi.de

[40] http://rdf2go.semweb4j.org

können gleiche Eigenschaften haben (zum Beispiel sneq:hasName). Die Regeln wurden abgewandelt, um Instanzen im SNEQ-Datenmodell auf Gleichheit bzw. Ähnlichkeit zu prüfen.

R1: Wenn die 'Labels' gleich sind, dann sind wahrscheinlich die Entitäten gleich
'Labels' sollten verglichen werden, als 'Labels' werden hier auch die Werte der Eigenschaften aufgefasst, wie zum Beispiel Namen und Interessen.

R13: Zwei Instanzen sind mit der gleichen Eigenschaft zu einer anderen Instanz verbunden, dann sind die zwei Instanzen ähnlich
Regel kann übernommen werden.

R15: Wenn zwei Klassen, zwei Eigenschaften oder zwei Instanzen mit bestimmter Eigenschaft, die die Gleichheit ausdrückt, verbunden sind, dann sind sie gleich.
Regel kann für Instanzen übernommen werden.

Ein vereinfachter Algorithmus sieht folgendermaßen aus: Im ersten Schritt werden die 'Labels' in einem oder mehreren RDF-Modellen verglichen. Es wird überprüft, ob es bereits gleiche Instanzen in der Datenmenge gibt (R15). Die Instanzen mit gleichen 'Labels' werden gemerkt. Im zweiten Schritt können Instanzen nach R13 verglichen werden, um weitere Gleichheiten festzustellen. Die gleichen Instanzen werden mit der Eigenschaft aus R15 verbunden. Im dritten Schritt werden die Schritte eins und zwei solange wiederholt, bis es keine signifikanten Änderungen in der bereits erkannten Menge mehr gibt.

3.8.1 Algorithmus

Der konkrete Entwurf für die Feststellung der Gleichheit zwischen Personen aus unterschiedlichen Netzwerken ist in der Abbildung 18 dargestellt. R15 wurde im Algorithmus nicht verwendet, denn man führt die Erkennung auf noch unverarbeiteten extrahierten Daten aus den Netzwerken durch. Die zu vergleichenden 'Labels' wurden so gewählt, dass die Erkennung ohne Wiederholungen von Schritt eins und zwei funktioniert.

Die richtige Funktionalität der nachfolgend beschriebenen Methode ist folgendermaßen begründet. Eine Person, die sich in verschiedenen Netzwerken anmeldet, gibt meistens überall den gleichen Namen an. Die kleinsten

Abweichungen werden durch den gewählten Grenzwert 2 der Levenshtein-Distanz berücksichtigt. Es ist höchst unwahrscheinlich, dass eine Person mehrere Freunde mit gleichen oder nahezu gleichen Namen hat.

```
1:  BEGIN Duplikaterkennung
2:    rdfModellNetzwerk1.ladeDatei('netzwerk1daten.ttl');
3:    rdfModellNetzwerk2.ladeDatei('netzwerk2daten.ttl');
4:    IDsundNamenNetzwerk1 = rdfModellNetzwerk1.ladeIDsundNamen();
5:    IDsundNamenNetzwerk2 = rdfModellNetzwerk2.ladeIDsundNamen();
6:    FOR EACH N1Person IN IDsundNamenNetzwerk1
7:      FOR EACH N2Person IN IDsundNamenNetzwerk2
8:        IF levenshtein(N1Person.Name, N2Person.Name) < 2)
9:        THEN gleicheNamen.add(N1Person.ID, N2Person.ID)
10:   FOR EACH gleicherName IN gleicheNamen
11:     FOR EACH N1Person
                IN rdfModellNetzwerk1.ladeKontakte(gleicherName.ID0)
12:       FOR EACH N2Person
                  IN rdfModellNetzwerk2.ladeKontakte(gleicherName.ID1)
13:         IF ( (N1Person.ID, N2Person.ID) IN gleicheNamen OR
                (N2Person.ID, N1Person.ID) IN gleicheNamen)
14:         THEN result.add(gleicherName.ID0, gleicherName.ID1)
15:              result.add(N1Person.ID, N2Person.ID)
16:  END Duplikaterkennung
```

Abbildung 18: Algorithmus in Pseudo-Code

In der Initialisierungsphase (Zeile 2-3) werden Daten im SNEQ-Datenformat in zwei RDF-Modelle geladen. Das Laden kann sowohl aus dem lokalen Speicher als auch direkt aus den Netzwerken erfolgen. Im ersten RDF-Modell stehen Daten aus einem Netzwerk, im zweiten Daten aus einem anderen Netzwerk.

In der Vorverarbeitungsphase (Zeile 4-5) werden aus beiden Modellen alle Personen mit ihren Namen in unterschiedliche Behälter geladen.

Im nächsten Schritt werden alle Namen der Personen aus einem Netzwerk mit den Namen der Personen aus dem anderen Netzwerk auf Gleichheit überprüft (Zeile 6-9). Diese Überprüfung erfolgt durch die Berechnung der im Abschnitt 2.5.1 beschriebenen Levenshtein-Distanz. Bevor die Distanz berechnet wird, werden die zu vergleichenden Namen vor der Berechnung vollständig in Kleinbuchstaben transformiert. Das geschieht damit die Levenshtein-Distanz von 'Peter' und 'peter' mit 0 berechnet wird, da es sich hier zweifellos um den gleichen Namen handelt. Außerdem werden bei den Namen die Umlaute durch 'ae', 'oe' und 'ue' ersetzt. Die Namen mit der Distanz von 1 werden als gleich markiert. Die IDs der Personen mit den gleichen Namen werden als Paare in einer Hash-Tabelle abgelegt.

Bei der Auswertung (Zeile 10-15) werden zu den Personen mit den als gleich erkannten Namen die Kontaktlisten aus den RDF-Modellen geladen. Es wird über alle Personen mit gleichen Namen und über alle ihren Kontakte iteriert. Es findet eine Überprüfung statt, ob die Namen der Kontakte aus unterschiedlichen Netzwerken gleich sind (Zeile 13). Hierzu müssen sie in der zuvor aufgestellten Hash-Tabelle aller namensgleichen Paare als Paar enthalten sein. Ist das der Fall, werden beide Personen mit ihren zwei Profilen aus unterschiedlichen Netzwerken als gleich erkannt und dem Ergebnis hinzugefügt (Zeile 14-15).

3.9 Architektur

Bei dem Entwurf der Architektur des Systems wurde darauf geachtet, dass die Erweiterbarkeit gewährleistet wird. Es sollte problemlos möglich sein, die Unterstützung weiterer Netzwerke zu implementieren.

Das System ist wie folgt aufgebaut (siehe Abbildung 19). Es gibt eine abstrakte Oberklasse. Für jedes ausgewählte soziale Online-Netzwerk existiert eine konkrete 'Network'-Klasse. Diese sind von der abstrakten Oberklasse abgeleitet. Diese Klassen sind die Schnittstellen zum jeweiligen Netzwerk. Die Klassen haben folgende Funktionen.

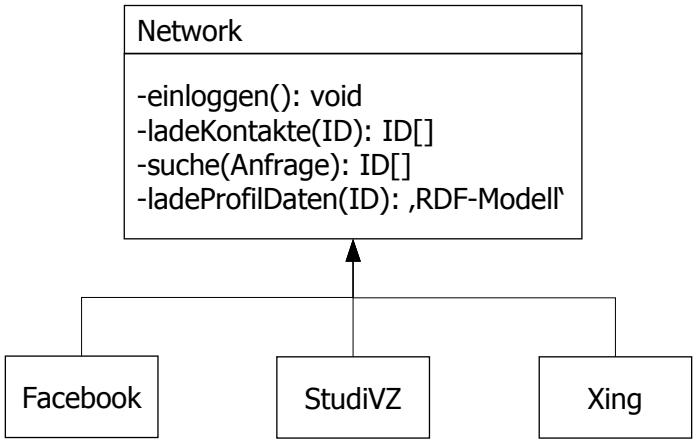

Abbildung 19: Architektur

Es gibt eine Methode zum **Einloggen** in das Netzwerk. Die HTTP-Zugriffskomponente verwendet das vom Nutzer bei der Erstellung des 'Network'-Objekts übergebene Passwort und den Benutzernamen.

Die Methode **ladeKontakte** liefert die Freunde eines bestimmten Netzwerkteilnehmers. Die ID dieses Netzwerkteilnehmers wird als Parameter beim Aufruf übergeben. In jedem Netzwerk bekommt jeder Teilnehmer eine einzigartige ID zwecks eindeutiger Identifikation innerhalb des Netzwerks. Diese besteht entweder aus beliebiger Kombination von Buchstaben und Zahlen, wie das zum Beispiel bei Facebook der Fall ist, oder sie besteht aus dem Namen und Vornamen des Nutzers und gegebenfalls einer Zahl am Ende, falls man nicht der erste mit dieser Namenskombination im Netzwerk ist, wie das bei Xing ist. Diese Funktion liefert einen Behälter. Darin enthalten sind die IDs der Freunde von der Person mit der ID des gegebenen Parameters.

Die Methode **suche** ruft die Suche im entsprechenden Netzwerk auf. Der Parameter ist eine Zeichenkette. Die Methode liefert einen Behälter. Darin enthalten sind die IDs der Personen, die die Ergebnisse der Suchanfrage sind.

Die Methode **ladeProfilDaten** extrahiert die benötigten Profildaten aus dem entsprechenden Netzwerk und lädt diese in ein RDF-Modell. Die Parameter sind eine ID eines Teilnehmers und eine Referenz auf ein RDF-Modell. Die gelesenen Daten sind die Daten zu der übergebenen ID.

Die Übergabe der Freunde eines Teilnehmers wurde in einer separaten Methode realisiert. Der Vorteil dabei ist, man kann bei Bedarf nur nach Freundschaften das Netzwerk durchforsten und speichert nur die Beziehungen und die Namen der Teilnehmer. Die Profilseiten werden nicht eingelesen, falls dafür kein Bedarf besteht.

4 Implementierung

Die Implementierung des Systems erfolgte in Java. Es wurden alle Schnittstellen zu den im Abschnitt 3.3 ausgewählten sozialen Online-Netzwerken realisiert.

In diesem Kapitel werden die wichtigsten Details der Implementierung der einzelnen Methoden beschrieben. Die Methoden in den Adaptern für die unterschiedlichen sozialen Online-Netzwerke haben die gleichen Funktionalitäten. Die folgenden Abschnitte beschreiben jeweils eine Methode.

Der Quellcode des Systems wird unter *http://sneq.googlecode.com* zur Verfügung gestellt.

4.1 Einloggen

Das Einloggen findet mit Hilfe einer Instanz der im Abschnitt 3.6.1 gewählten HTTP-Zugriffskomponente statt. Die HTTP-Zugriffskomponente benötigt zum Einloggen einen Benutzernamen und ein Passwort. Sowohl der Benutzername als auch das Passwort werden bei der Erstellung einer 'Network'-Instanz (siehe Abschnitt 3.9) übergeben. Diese Werte werden internen Variablen zugewiesen. Es wird eine HTTP-Post-Anfrage erzeugt. Der Benutzername und das Passwort werden als Parameter der HTTP-Post-Anfrage gesetzt. Der Benutzername ist entweder eine Email-Adresse, wie das bei Facebook und StudiVZ der Fall ist, oder ein Nickname wie bei Xing.

Neben diesen zwei vom Nutzer des Systems abhängigen Parametern gibt es unterschiedlich vom Netzwerk mehrere statische und generische HTTP-Parameter, die für ein erfolgreiches Einloggen gebraucht werden. Statische Parameter haben bei jedem Einloggen den gleichen Wert, generische Parameter können bei jedem Einloggen verschieden sein.

Die statischen Parameter beim Einloggen in die jeweilige Netzwerke sind:

- Facebook:

  `md5pass` mit leerem String als Wert

  `noerror` mit dem Wert '1'

  `doquicklogin` mit dem Wert 'Login'

- StudiVZ:

  `noerror` mit dem Wert '1'

  `doquicklogin` mit dem Wert 'Einloggen'

- Xing:

  `op` mit dem Wert 'login'

  `dest` mit dem Wert '/app/user?op=home'

Die generischen Parameter sind:

- Facebook: `charset`, `_test`.
- StudiVZ: keine generischen Parameter.
- Xing: `_sid`.

Die generischen Parameter werden auf die folgende Weise erhalten: Es wird auf die Webseite mit der Einloggen-Schnittstelle mit der HTTP-Zugriffskomponente eine HTTP-Get-Anfrage ausgeführt. Die vom Server zurückgegebene Webseite wird mit Cyberneko-HTML-Parser[41] geparst. Ein XML-Reader von dom4j[42] transformiert sie in ein XML-Dokument. Aus diesem XML-Dokument wird auf die erforderlichen Werte mit XPATH zugegriffen.

Beispiel: `_sid: '//*[name()='INPUT' and ./@name='_sid']'`. Der erhaltene Knoten hat ein Attribut 'value', wo sich der erforderliche Wert befindet. Der dazugehörige HTML-Auszug ist in der Abbildung 20 dargestellt.

4.2 Extrahieren der Kontakte

Diese Methode benötigt eine ID innerhalb des Netzwerkes als Parameter. Zu dieser ID gibt die Methode eine Menge der IDs zurück, die den Freunden der Parameter-ID gehören. Die Informationen sind nur dann vorhanden, wenn die Freunde der bestimmten Person für den eingeloggten Nutzer sichtbar sind. Sind diese Informationen für den eingeloggten Nutzer nicht verfügbar, wird eine leere Menge zurückgegeben.

Die Methode verwendet die HTTP-Zugriffskomponente, die in das Netzwerk eingeloggt sein muss. Es wird eine HTTP-Get-Anfrage an die Seite mit der Freundesliste der bestimmten Person geschickt. Falls die Freunde der Person nicht alle auf eine Seite passen, werden auf die nächsten Seiten der Freundesliste HTTP-Get-Anfragen ausgeführt, so dass man diese Informationen so vollständig wie möglich bekommt. Die IDs der Freunde werden aus den Webseiten extrahiert.

Es wurde eine weitere Methode zum Extrahieren der Freunde implementiert. Diese gibt eine HashMap zurück. In dieser HashMap sind zusätzlich zu

[41] http://sourceforge.net/projects/nekohtml

[42] http://www.dom4j.org

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de" lang="de">
...
<body >
...
<form action="https://www.xing.com/app/user" method="post"
    name="loginform">
<input type="hidden" name="op" value="login"/>
<input type="hidden" name="dest" value="/app/user?op=home"/>
<input type="hidden" name="_sid" value="0.2ae0cd"/>
<label for="login_user_name" class="b">Benutzername</label><br/>
<input type="text" class="text" tabindex="1" name="login_user_name"
    id="login_user_name" maxlength="80" style="width:99%;" /><br/>
<label for="login_password" class="b">Passwort</label><br/>
<input type="password" class="text" tabindex="2" name="login_password"
    id="login_password" maxlength="25" style="width:99%;" />
<br/><p class="mp0">
<input class="fl" type="checkbox" name="perm" id="perm" tabindex="3" />
...
</body >
</html>
```

Abbildung 20: HTML-Auszug

den IDs auch die Namen enthalten. Die IDs sind die Schlüssel, die Namen sind die dazugehörigen Werte.

4.3 Suche

Die Suchmethode benötigt einen String als Parameter. Dieser String ist die gewünschte Suchanfrage. Die Suchanfrage wird an die Suchschnittstelle des Netzwerkes mit der HTTP-Zugriffskomponente übermittelt. Die Ergebnisse werden aus den vom Netzwerkserver zurückgegebenen Webseiten extrahiert. Es wird eine Menge der IDs erzeugt und als Rückgabewert an den Aufrufer der Methode zurückgegeben.

Es gibt eine erweiterte Version dieser Methode. Statt einer Menge der IDs wird eine HashMap zurückgegeben, die sowohl die IDs als auch die zu diesen IDs gehörenden Namen enthält. Die IDs sind die Schlüssel, die Namen sind die dazugehörigen Werte.

4.4 Extrahieren der Profildaten

Die Parameter dieser Methode sind eine Referenz auf ein RDF-Modell und eine bestimmte ID. Diese ID bestimmt die Person aus dem Netzwerk, von welcher die Profilinformationen extrahiert werden. Auch hier können nur Daten gelesen werden, die für den mit der HTTP-Zugriffskomponente eingeloggten Nutzer sichtbar sind.

Die Webseiten mit den erforderlichen Daten werden mit der HTTP-Zugriffskomponente mit HTTP-Get-Anfragen angefordert. Diese Webseiten werden mit Cyberneko-HTML-Parser geparst und von einem dom4j-XML-Reader in XML transformiert. Die Daten werden mit XPATH-Ausdrücken aus dem XML gelesen. Die erhaltenen Informationen werden in das gegebene RDF-Modell geladen. Dieser Ablauf ist in der Abbildung 21 dargestellt.

Die Methode existiert in drei Ausführungen. Die erste lädt nur die persönlichen Daten ohne die Beziehungen zu den Freunden in das RDF-Modell. Die zweite lädt nur die IDs der Kontakte und die Beziehungen der Kontakte zu der gegebenen ID. Die dritte ist eine Erweiterung der zweiten. Zusätzlich werden die Namen der Freunde in das RDF-Modell geladen.

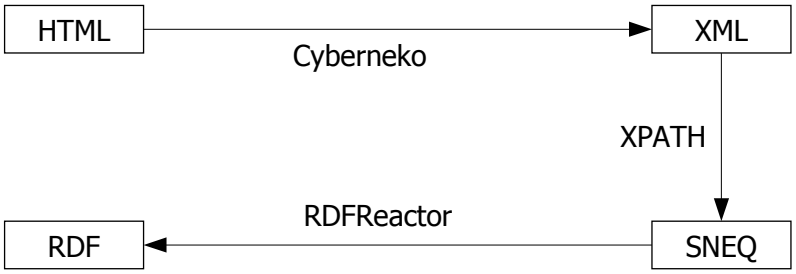

Abbildung 21: Von HTML nach RDF

4.5 Besonderheiten bei der Implementierung

Um die richtige Funktionalität der implementierten Methoden zu gewährleisten, wurden mit der JUnit-Bibliothek Testmethoden implementiert.

Bei der Implementierung der Datenextraktion für StudiVZ wurde festgestellt, dass StudiVZ Eingabeaufforderungen an den eingeloggten Nutzer stellt. Diese bestehen aus Bildern mit Zahlen und Buchstabenkombinationen, die eingegeben werden müssen, um weiter die Profile betrachten zu können. Das implementierte System kann dieses Problem im Moment nicht automatisch lösen. Das System erkennt, dass eine Eingabeaufforderung von StudiVZ

gestellt wurde, macht eine Ausgabe und wartet bis es weitergehen soll. Der Nutzer kann sich im Browser einloggen, die Eingabeaufforderung erledigen und dem System mitteilen, dass es weitergehen kann.

4.6 Anwendungsbeispiele

In diesem Abschnitt werden zwei Anwendungsbeispiele mit dem dazugehörigen Programmcode beschrieben.

Extrahieren der Kontakte bis zu einem beliebigen Grad

Die Abbildungen 22 und 23 zeigen den Programmcode, wie man die Kontakte beliebigen Grades aus einem Netzwerk laden kann.

Bei der Initialisierung werden Benutzername und Passwort gesetzt. Das entsprechende 'Network'-Objekt wird erzeugt. Die Methode 'Einloggen' wird aufgerufen. Ein RDF-Model wird geladen und geöffnet. Eine 'sneq:Person' wird erzeugt, deren Kontakte geladen werden sollen. Es wird festgelegt, bis zu welchem Grad die Kontakte geladen werden sollen. '2' bedeutet: es werden direkte Kontakte und die direkten Kontakte dieser Kontakte geladen. Die Rekursion wird durch den Methodenaufruf mit entsprechenden Parametern gestartet.

Es wird überprüft, ob der Kontakt bereits geladen wurde. Es wird überprüft, wie weit man bei der Rekursion bereits abgestiegen ist. Es werden die IDs der Kontakte zu der entsprechenden ID geladen. Die Kontakte werden im RDF-Modell angelegt. Die Eigenschaft 'hasFriend' für die Freundschaftliche Beziehung zwischen den Personen wird angelegt. Es wird überprüft, ob die Daten zu diesem Profil bereits extrahiert wurden. Man steigt mit der ID dieser Person in der Rekursion ab und lädt die Kontakte ein Grad kleiner.

Suche nach einem bestimmten Namen in einem Netzwerk und lokales Speichern der gefundenen Profile im SNEQ-Format:

Die Abbildung 24 zeigt, wie man die Profildaten bestimmter Personen aus einem Netzwerk lädt und diese Daten dann lokal speichert.

Bei der Initialisierung werden Benutzername und Passwort gesetzt. Das entsprechende 'Network'-Object wird erzeugt. Die Methode 'Einloggen' wird aufgerufen. Ein RDF-Model wird geladen und geöffnet. Die Suche mit der Anfrage 'peter meier' wird aufgerufen. Über die Ergebnisse der Suche wird iteriert. Für jede gefundene Person werden die Profildaten und die Kontakte in das RDF-Modell geladen.

```
static String XING = "http://www.xing.com/profile/";

public static void main(String[] args) {
  //initialisiere Benutzername und Passwort
  String benutzername = "hans.schmidt";
  String passwort = "schmidtspasswort";
  //erzeuge ein Network-Xing-Object
  Network network = new Xing(benutzername, passwort);
  //einloggen in das Netzwerk
  network.login();
  //erzeuge und öffne ein RDF-Modell für die Daten
  Model m = RDF2Go.getModelFactory().createModel();
  m.open();
  //initialisiere die ID des Nutzers zu der die Kontakte
  //extrahiert werden sollen
  String id = "Hans_Schmidt";
  //erzeuge den Nutzer im RDF-Modell
  Person p = new Person(m, "http://www.xing.com/profile/" + name);
  //initialisiere den Grad der Kontakte, die extrahiert werden sollen
  int grad = 2;
  //starte die Rekursion mit den entsprechenden Parametern
  crawlFriends(grad, name, network, m, p);
  //schreibe die extrahierten Daten mit richtigem Encoding
  //in den lokalen Speicher
  Writer writer = null;
  FileOutputStream fos = new FileOutputStream(
                            ""./target/hansfriendsofriends.ttl"");
  writer = new OutputStreamWriter(fos, Charset.forName("UTF-8"));
  m.writeTo(writer, Syntax.Turtle);
}
```

Abbildung 22: Anwendungsbeispiel 1 in Java - Initialisierung

Ein Objekt zum lokalen Speichern wird mit den entsprechenden Einstellungen des Speicherortes und der Zeichenkodierung erzeugt. Das RDF-Modell wird angewiesen, mit Hilfe dieses Objekts die geladenen Daten in der entsprechenden RDF-Darstellung abzuspeichern.

```
static Set<String> known = new HashSet<String>();

public static void crawlFriends(int levels, String personId,
Network network, Model model, Person p) {
  //falls die Kontakte zu dieser ID schon extrahiert wurden, abbrechen
  if (known.contains(personId)) return;
  //falls man bei Grad 0 angekommen ist, abbrechen
  if (levels == 0) return;
  //extrahiere Kontakte in eine Liste
  Collection<String> list = network.getFriends(personId);
  // iteriere über die Kontaktliste
  for (String friendId : list) {
    //erzeuge die Kontaktperson im RDF-Modell
    Person friendPerson = new Person(model, XING + friendId);
    //erzeuge die Beziehung zwischen Person und Kontaktperson
    p.addHasFriend(friendPerson);
    //merke die Person, deren Kontakte extrahiert wurden
    known.add(personId);
    //extrahiere die Kontakte vom Grad-1 von der Kontaktperson
    crawlFriends(levels-1, friendId, network, model, friendPerson);
  }

}
```

Abbildung 23: Anwendungsbeispiel 1 in Java - Rekursion

```
//initialisiere Benutzername und Passwort
String benutzername = "hans.schmidt@fzi.de";
String passwort = "schmidtspasswort";
//erzeuge ein Network-Facebook-Object
Network network = new Facebook(benutzername, passwort);
//einloggen in das Netzwerk
network.login();
//erzeuge und öffne ein RDF-Modell für die Daten
Model m = RDF2Go.getModelFactory().createModel();
m.open();
//aufrufen der Netzwerksuche mit der Suchanfrage als Parameter
Collection<String> col = network.search("peter meier");
//iteriere über die Suchergebnisse
for(String s: col) {
  //extrahiere die Profildaten in das RDF-Modell zur gegebener ID
  network.fillRdfModel(s, m);
  //extrahiere die Profilbeziehungen in das RDF-Modell zur gegebener ID
  network.fillRdfModelWithFriends(s, m);
}
//schreibe die extrahierten Daten mit richtigem Encoding
//in den lokalen Speicher
Writer writer = null;
FileOutputStream fos = new FileOutputStream(
                            "./target/search_petermeier.ttl");
writer = new OutputStreamWriter(fos, Charset.forName("UTF-8"));
m.writeTo(writer, Syntax.Turtle);
```

Abbildung 24: Anwendungsbeispiel 2 - Suchen und Speichern

5 Evaluation

In diesem Kapitel werden die Ergebnisse der Evaluation präsentiert. Im ersten Teil gibt es eine Bewertung des Systems bezüglich der im Abschnitt 3.2 an das System gestellten Anforderungen. Im zweiten Teil wird das System überprüft, inwieweit es hilft, die im Abschnitt 1.1 beschriebenen Probleme zu lösen.

5.1 Das System

Anforderung 1: Vollständiges Einlesen aller für den Nutzer sichtbaren Daten. Die Abschnitte 3.4 und 3.5 haben gezeigt, welche Datenvielfalt in sozialen Online-Netzwerken herrscht. Die Architektur des Systems erlaubt es, alle für den Nutzer sichtbaren Daten aus den sozialen Online-Netzwerken zu extrahieren. Die Implementierung eines Systems, die alle Daten extrahieren kann, ist sehr zeitintensiv. Es wurde beschlossen, vorerst nur das Einlesen einiger wichtiger persönlicher Daten zu implementieren. Diese Daten sind der Name, die freundschaftlichen Beziehungen, das Geburtsdatum, die Adresse und die Interessen. Abhängig vom Netzwerk funktioniert auch das Einlesen der besuchten Bildungseinrichtungen, der aktuellen Arbeitgeber und des Geschlechts. Die im Abschnitt 3.6 beschriebene Einlesemethode erlaubt es aber, mit entsprechendem Zeiteinsatz und Programmieraufwand das vollständige Einlesen zu implementieren.

Anforderung 2: Speichern der extrahierten Daten in einem einheitlichen Datenmodell. Das entworfene einheitliche Datenmodell kann alle nutzerrelevanten Daten aufnehmen. Das System speichert die aus den Netzwerken extrahierten Daten im entworfenen einheitlichen Datenmodell (SNEQ). Die Daten liegen strukturiert vor. Der Zugriff auf die Daten gestaltet sich problemlos (Abschnitt 3.7).

Anforderung 3: Suche nach Personen in verschiedenen Netzwerken. Die Suche nach Teilnehmern in den ausgewählten Netzwerken (Abschnitt 3.3) über das System funktioniert wie erwünscht. Die Implementierung simuliert die einfache Suche in allen ausgewählten Netzwerken. Sie reicht vollkommen aus, um nach Personen mit bestimmten Namen zu suchen. Die Suche wurde mit verschiedenen Suchanfragen getestet. Es wurden von dem System die gleichen Ergebnisse zurückgegeben, die auch im Browser nach der Eingabe dieser Suchanfragen sichtbar waren. Eine Ausnahme bildet hier die Suche in Facebook. Auch im

Browser kann sie bei wiederholter Eingabe identischer Suchanfragen verschiedene Ergebnisse zurückgeben.

Anforderung 4: Erkennung der Profile von gleichen Personen aus unterschiedlichen Netzwerken. Die im Abschnitt 3.8 beschriebene Erkennung der Profile von gleichen Personen aus unterschiedlichen Netzwerken wurde implementiert und getestet. Zum Testen wurde die Schnittstelle des Systems zur Suche aufgerufen. Es wurde nach drei bestimmten Namen in allen drei Netzwerken gesucht. Von diesen drei Namen wusste man, dass es gleiche Personen mit diesen Namen in allen drei Netzwerken gibt. Die Abbildung 25 zeigt, wieviele Suchergebnisse es in den Netzwerken zu den Anfragen gab. Die Ergebnisse wurden mit ihren Kontakten im SNEQ-Format lokal gespeichert. Die Analyse der Daten erfolgte mit dem im Abschnitt 3.8 entworfenen Algorithmus. In der Abbildung 26 sind die aufsummierten Anzahlen der aus den einzelnen Netzwerken extrahierten Profile dargestellt. Die Profile wurden jeweils nur mit IDs, Vor- und Nachnamen extrahiert, da weitere Daten für die Analyse nicht benötigt werden. Die Analyse der Daten erfolgte jeweils für zwei Netzwerke. Die Analyse zwischen Facebook und StudiVZ ergab 18, zwischen Facebook und Xing 27, zwischen StudiVZ und Xing 23 Treffer. Die dabei erkannten Treffer wurden manuell im Browser in den Netzwerken überprüft. Es konnten keine falschen Treffer bei den als gleich erkannten Personen festgestellt werden. Es wurde nicht untersucht, ob alle Profile gleicher Personen erkannt wurden. Die Profile der drei Personen, von denen man sicher wusste, dass es sich um gleiche Personen handelt, wurden mit allen ihren Freunden richtig erkannt.

	Facebook	StudiVZ	Xing
Anfrage 1	8	33	12
Anfrage 2	2	4	4
Anfrage 3	1	3	1

Abbildung 25: Anzahl der Ergebnisse für die Suchanfragen

	Facebook	StudiVZ	Xing
Anzahl	138	1906	505

Abbildung 26: Gesamtanzahlen der extrahierten Profile

5.2 Profildaten

Im Anhang B befindet sich eine exemplarische Aggregation der Profildaten einer Person aus drei sozialen Online-Netzwerken.

5.3 Anwendung

Im Folgenden gibt es eine Überprüfung, in welchem Umfang es möglich ist, mit dem System die im Abschnitt 1.1 beschriebenen Probleme zu lösen.

Problem 1: Suche nach Freunden in verschiedenen Netzwerken. Man kann in allen unterstützten sozialen Online-Netzwerken nach seinen noch nicht zur Freundesliste hinzugefügten Freunden mit dem entworfenen und implementierten System suchen. Das System kann die Ergebnisse der Suche mit seinen Freundeslisten nach gleichen Personen überprüfen. Ob die erkannten gleichen Personen sich bereits in den jeweiligen Freundeslisten befinden, kann leicht festgestellt werden.

Problem 2: Nutzer haben keinen Überblick über ihre Kontakte aus verschiedenen Netzwerken. Die automatische Extraktion der Daten aus den unterstützten Netzwerken und die Aggregation dieser Daten in das entworfene einheitliche Datenmodell wird vom System unterstützt. Aus diesen Daten kann eine Gesamtübersicht generiert werden. Die Daten können problemlos aus dem einheitlichen Format in eine Textdarstellung überführt werden. Um die aktuellen Daten für die Gesamtübersicht zu erhalten, braucht man nur einmal das System zu starten. Das manuelle Anmelden in die Netzwerke und das manuelle Stöbern nach Daten entfällt.

Problem 3: Auflistung redundanter Kontakte eines Nutzers aus allen Netzwerken. Da man eine Gesamtübersicht eigener Kontakte mit dem System erhalten und die Erkennung gleicher Personen aus unterschiedlichen sozialen Online-Netzwerken auf diesen Daten mit dem System durchführen kann, ist es nun möglich, beliebige Kombinationen der Vereinigungs- und der Schnittmengen eigener Freunde aus unterschiedlichen Netzwerken zu bilden. Somit ist es möglich, automatisch zu erkennen, dass man zum Beispiel mit vier Netzwerken schon alle seine Kontakte abdeckt und das fünfte gar nicht braucht.

Für den praktischen Einsatz sollte allerdings die Unterstützung weiterer Netzwerke implementiert werden, da diese Art von Analysen die Personen, die in vielen Netzwerken angemeldet sind, mehr interessieren wird, als die Personen, die nur in zwei oder drei Netzwerken ein Konto haben.

Problem 4: lokales Speichern der Daten seiner Freunde. Es kann eine Gesamtübersicht der eigenen Freunde mit den persönlichen Daten aus allen unterstützten Netzwerken extrahiert werden. Die Gesamtübersicht liegt strukturiert vor. Diese Daten können in andere Formate portiert werden. Es können beliebige Transformationsmöglichkeiten implementiert werden, die diese Gesamtübersicht zum Beispiel in das Datenformat des eigenen Adressbuchs transformieren. Auch können diese Daten im 'Social Semantic Desktop' von NEPOMUK [GHM+07] verwendet werden.

Problem 5: Berechnung des 'sozialen Kapitals' des eigenen Profils und der eigenen Freunde. In Zusammenarbeit mit SNA-Forschern kann eine Bewertung des 'sozialen Kapitals' der Nutzerkonten implementiert werden. Das System wurde so entworfen, dass es alle in den Netzwerken sichtbaren Daten extrahieren kann. Die Profildaten der eigenen Kontakte sind grundsätzlich in allen Netzwerken sichtbar. Die Berechnung des 'sozialen Kapitals' kann somit durchgeführt werden.

Problem 6: einfache Fragestellungen von Forschern der sozialen Netzwerke. Die exemplarisch im Abschnitt 1.1 angegebenen Fragen können prinzipiell beantwortet werden. Die Daten zur Durchführung dieser Analysen können aus den Netzwerken gelesen werden. Die verschiedenen Abwehrmechanismen der unterschiedlichen Netzwerke zum Schutz ihrer Nutzer sollten beim Datensammeln nicht außer Acht gelassen werden.

6 Zusammenfassung

In der vorliegenden Arbeit wurden Probleme für die einzelnen Nutzer sozialer Online-Netzwerke und Fragestellungen der Netzwerkforscher beschrieben. Es wurden einige der beliebtesten sozialen Online-Netzwerke vorgestellt und die Grundlagen erläutert, auf denen diese Arbeit aufbaut.

Aus den beliebtesten Netzwerken wurden drei ausgewählt. Die in diesen drei Netzwerken vorkommenden Profildaten der Teilnehmer wurden analysiert. Aufgrund dieser Analysen wurden die Datenmodelle für die drei Netzwerke erstellt. Aus diesen drei Datenmodellen wurde ein einheitliches Datenmodell (SNEQ-Datenmodell) entworfen. Das SNEQ-Datenmodell ist leicht erweiterbar und vereinheitlicht die Datenmodelle der drei ausgewählten sozialen Online-Netzwerke. Das SNEQ-Datenmodell deckt vollständig alle in den drei ausgewählten Netzwerken vorkommenden sichtbaren Daten ab.

Aus den Problemen und Fragestellungen wurden Systemanforderungen aufgestellt. Das System wurde entworfen und implementiert. Aufgrund der Architektur des erstellten Systems kann sie alle sichtbaren Daten aus den sozialen Online-Netzwerken extrahieren und sie in das einheitliche Datenmodell aggregieren. Außerdem ist das System für beliebig viele Netzwerke erweiterbar. Auch wurde eine Erkennung der Profile von gleichen Personen aus unterschiedlichen Netzwerken entworfen und implementiert.

Es erfolgte eine Evaluierung des Systems. Hier wurde festgestellt, dass das System die aufgestellten Anforderungen erfüllt. Das System kann prinzipiell alle sichtbaren Profildaten aus den Netzwerken extrahieren, die Daten im SNEQ-Datenmodell speichern, die Suchschnittstellen der einzelner Netzwerke verwenden und auf den extrahierten Daten eine Erkennung der Profile von gleichen Personen durchführen. Damit eignet sich das SNEQ-System zur Lösung der in 1.1 beschrieben Probleme. Mit dem SNEQ-System wurde die Suche nach Freunden aus einem Netzwerk in mehreren anderen Netzwerken automatisiert und die Suche nach Profilinformationen vereinfacht. Das System ist in der Lage, redundante Kontakte aus verschiedenen Netzwerke zu erkennen und die extrahierten Daten im SNEQ-Datenmodell zu speichern, was Transformationen in beliebige Datenformate erleichtert. Mit Hilfe des Systems können das 'soziale Kapital' der Profile berechnet und verschiedene Fragestellungen der Netzwerkforscher beantwortet werden.

6.1 Ausblick

Es sind verschiedene Erweiterungen und Einsatzmöglichkeiten des Systems vorstellbar. So kann das System um die Unterstützung weiterer Netzwerke erweitert werden. Eine Erweiterung des Systems mit Schreibfunktionen ist möglich, um eigene Profildaten in allen unterstützten Netzwerken automatisch zu aktualisieren.

Verschiedene Transformationsmöglichkeiten können mit Hilfe des einheitlichen Datenmodells realisiert werden. Die Daten können zum Beispiel in FoaF oder in das Format anderer Adressbücher transformiert werden.

Es können weitere Möglichkeiten der Erkennung gleicher Personen aus verschiedenen Netzwerken implementiert werden. Man kann zum Beispiel die Namen, die Bildung und die Arbeitstellen vergleichen und ist nicht auf die Freundeslisten angewiesen.

Die Überprüfung der Namen auf Gleichheit kann auf andere Weisen erfolgen. In [EIV07] findet man eine Übersicht über die verschiedenen Möglichkeiten, Zeichenketten zu vergleichen und Duplikaterkennung zu optimieren.

A SNEQ-Schema

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix sneq: <http://www.xam.de/2007/sneq#> .


######
## social network class with property
######


sneq:SocialNetwork a rdfs:Class
; rdfs:label "SocialNetwork"
; rdfs:comment "Each person can belong to many social networks"
.

sneq:hasSocialNetwork a rdf:Property
; rdfs:label "hasSocialNetwork"
; rdfs:comment "Each person can belong to many social networks"
; rdfs:domain sneq:Person
; rdfs:range sneq:SocialNetwork
.

sneq:facebook a sneq:SocialNetwork .
sneq:xing a sneq:SocialNetwork .
sneq:studivz a sneq:SocialNetwork .

sneq:usedByNetwork a rdf:Property
; rdfs:label "usedByNetwork"
; rdfs:domain rdf:resource
; rdfs:range sneq:SocialNetwork
.


######
## actor + subclasses person and group
######


sneq:Actor a rdfs:Class
; rdfs:label "Actor"
; rdfs:comment "an actor can be a person or a group"
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.
```

```
sneq:Person a rdfs:Class
; rdfs:label "Person"
; rdfs:comment "A person in a social network.
    id is stored as rdfs:label."
; rdfs:subClassOf sneq:Actor
.

sneq:Group a rdfs:Class
; rdfs:label "Group"
; rdfs:comment "A person can participate in groups."
; rdfs:subClassOf sneq:Actor
.


######
## group related classes
######


sneq:DiscussionBoard a rdfs:Class
; rdfs:label "DiscussionBoard"
; rdfs:comment "Discussion board of a group"
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:DiscussionTopic a rdfs:Class
; rdfs:label "DiscussionTopic"
; rdfs:comment "topic on Discussion board"
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


######
## timedobject + textobject
######


sneq:TimedObject a rdfs:Class
; rdfs:label "TimedObject"
; rdfs:comment "parent class for all timed items"
.

sneq:TimedTextObject a rdfs:Class
; rdfs:label "TimedTextObject"
; rdfs:comment "parent class for timed text items like comments"
; rdfs:subClassOf sneq:TimedObject
.
```

```
######
## photo + related classes
######

sneq:Photo a rdfs:Class
; rdfs:label "Photo"
; rdfs:comment "photo"
; rdfs:subClassOf sneq:TimedObject
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:PhotoAlbum a rdfs:Class
; rdfs:label "PhotoAlbum"
; rdfs:comment "photoalbum"
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.


######
## wall + minifeed + comment
######

sneq:Wall a rdfs:Class
; rdfs:label "Wall"
; rdfs:comment "wall"
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:MiniFeed a rdfs:Class
; rdfs:label "MiniFeed"
; rdfs:comment "minifeed"
; sneq:usedByNetwork sneq:facebook
.

sneq:MiniFeedStory a rdfs:Class
; rdfs:label "MiniFeedStory"
; rdfs:comment "blank node, story in mini-feed"
; rdfs:subClassOf sneq:TimedTextObject
; sneq:usedByNetwork sneq:facebook
.

sneq:Comment a rdfs:Class
; rdfs:label "Comment"
; rdfs:comment "blank node, message in wall, comment to a photo,
    post in Discussion topic"
; rdfs:subClassOf sneq:TimedTextObject
.
```

```
######
## other classes
######

sneq:Network a rdfs:Class
; rdfs:label "Network"
; rdfs:comment "A person can belong to networks (germany,
    uni karlsruhe, fzi), facebook, studivz only one"
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.

sneq:Website a rdfs:Class
; rdfs:label "Website"
; rdfs:comment "internet website"
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:Lection a rdfs:Class
; rdfs:label "Lection"
; rdfs:comment "A person can have many lections, studivz"
; sneq:usedByNetwork sneq:studivz
.

sneq:Education a rdfs:Class
; rdfs:label "Education"
; rdfs:comment "blank node for education"
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:Job a rdfs:Class
; rdfs:label "Job"
; rdfs:comment "blank node for Job"
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:Sex a rdfs:Class
;rdfs:label "Sex"
; sneq:usedByNetwork sneq:studivz, sneq:facebook .

sneq:sexMale a sneq:Sex .
sneq:sexFemale a sneq:Sex .

sneq:IMName a rdfs:Class
; rdfs:label "IMName"
; rdfs:comment "blank node for instant messaging accounts"
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.
```

```
sneq:IMService a rdfs:Class
; rdfs:label "IMService"
; rdfs:comment "each instant messaging name belongs to a service"
.

sneq:icq a sneq:IMService .
sneq:skype a sneq:IMService .
sneq:aim a sneq:IMService .
sneq:googletalk a sneq:IMService .
sneq:windowslive a sneq:IMService .
sneq:yahoo a sneq:IMService .
sneq:gadugadu a sneq:IMService .
sneq:jabber a sneq:IMService .
sneq:msn a sneq:IMService .

######
## group related properties
######

sneq:hasGroup a rdf:Property
; rdfs:label "hasGroup"
; rdfs:comment "each person can belong to groups"
; rdfs:domain sneq:Person
; rdfs:range sneq:Group
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasDiscussionBoard a rdf:Property
; rdfs:label "hasDiscussionBoard"
; rdfs:comment "each group has a Discussion board"
; rdfs:domain sneq:Group
; rdfs:range sneq:DiscussionBoard
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasDiscussionTopic a rdf:Property
; rdfs:label "hasDiscussionTopic"
; rdfs:comment "each Discussion board can have many topics"
; rdfs:domain sneq:DiscussionBoard
; rdfs:range sneq:DiscussionTopic
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasCategory a rdf:Property
; rdfs:label "hasCategory"
; rdfs:comment "category of the group"
; rdfs:domain sneq:Group
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook .
```

```
sneq:hasRecentInfos a rdf:Property
; rdfs:label "hasRecentInfos"
; rdfs:comment "recent informations"
; rdfs:domain sneq:Group
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.

sneq:hasOffice a rdf:Property
; rdfs:label "hasOffice"
; rdfs:comment "group can have an office"
; rdfs:domain sneq:Group
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.

######
## timedobject + timedtextobject related properties
######

sneq:hasTimeStamp a rdf:Property
; rdfs:label "hasTimeStamp"
; rdfs:comment "time message, photo added, minifeed story created"
; rdfs:domain sneq:TimedObject
; rdfs:range xsd:dateTime
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasText a rdfs:Property
; rdfs:label "hasText"
; rdfs:comment "each comment or mini-feed story can have a text"
; rdfs:domain sneq:TimedTextObject
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


######
## photo related properties
######

sneq:hasPhotoAlbum a rdf:Property
; rdfs:label "hasPhotoAlbum"
; rdfs:comment "each person can have photo albums"
; rdfs:domain sneq:Actor
; rdfs:range sneq:PhotoAlbum
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.
```

```
sneq:hasPhotoAlbumName a rdf:Property
; rdfs:label "hasPhotoAlbumName"
; rdfs:comment "each photolabum can have a name"
; rdfs:domain sneq:PhotoAlbum
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasPhoto a rdf:Property
; rdfs:label "hasPhoto"
; rdfs:comment "each person can have many photos,
    each photo can belong to photoalbum"
; rdfs:domain sneq:PhotoAlbum
; rdfs:range sneq:Photo
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasPALocation a rdf:Property
; rdfs:label "hasPALocation"
; rdfs:comment "each photoalbum can have a location"
; rdfs:domain sneq:PhotoAlbum
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.

sneq:hasPADescription a rdf:Property
; rdfs:label "hasPADescription"
; rdfs:comment "each photoalbum can have a description"
; rdfs:domain sneq:PhotoAlbum
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.

sneq:hasPhotoName a rdf:Property
; rdfs:label "hasPhotoName"
; rdfs:comment "each photo can have a name"
; rdfs:domain sneq:Photo
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:contains a rdf:Property
; rdfs:label "contains"
; rdfs:comment "who is in the photo"
; rdfs:domain sneq:Photo
; rdfs:range sneq:Person
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.
```

```
######
## wall + minifeed related properties
######

sneq:hasWall a rdf:Property
; rdfs:label "hasWall"
; rdfs:comment "each person or group can have a wall"
; rdfs:domain sneq:Actor
; rdfs:range sneq:Wall
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasMiniFeed a rdf:Property
; rdfs:label "hasMiniFeed"
; rdfs:comment "each person or group can have a minifeed"
; rdfs:domain sneq:Actor
; rdfs:range sneq:MiniFeed
; sneq:usedByNetwork sneq:facebook
.

sneq:hasStory a rdf:Property
; rdfs:label "hasStory"
; rdfs:comment "each mini-feed can have many stories"
; rdfs:domain sneq:MiniFeed
; rdfs:range sneq:MiniFeedStory
; sneq:usedByNetwork sneq:facebook
.

sneq:hasStoryKind a rdf:Property
; rdfs:label "hasStoryKind"
; rdfs:comment "kind of the story"
; rdfs:domain sneq:MiniFeedStory
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:facebook
.


######
## comment related properties
######

sneq:hasPhotoComment a rdf:Property
; rdfs:label "hasPhotoComment"
; rdfs:comment "each photo can have many comments"
; rdfs:domain sneq:Photo
; rdfs:range sneq:Comment
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.
```

```
sneq:hasWallComment a rdf:Property
; rdfs:label "hasWallComment"
; rdfs:comment "each wall can have many comments"
; rdfs:domain sneq:Wall
; rdfs:range sneq:Comment
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


sneq:hasTopicComment a rdf:Property
; rdfs:label "hasTopicComment"
; rdfs:comment "each Discussion topic can have many comments"
; rdfs:domain sneq:DiscussionTopic
; rdfs:range sneq:Comment
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


sneq:hasCommentAuthor a rdf:Property
; rdfs:label "hasCommentAuthor"
; rdfs:comment "author of the wall message"
; rdfs:domain sneq:Comment
; rdfs:range sneq:Person
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


sneq:hasMemoComment a rdf:Property
; rdfs:label "hasMemoComment"
; rdfs:comment "each person can give other persons memo,
    author wrote this memo"
; rdfs:domain sneq:Person
; rdfs:range sneq:Comment
; sneq:usedByNetwork sneq:xing
.


sneq:hasPersonalTagComment a rdf:Property
; rdfs:label "hasPersonalTagComment"
; rdfs:comment "each person can give other persons personal tag,
    author wrote this tag"
; rdfs:domain sneq:Person
; rdfs:range sneq:Comment
; sneq:usedByNetwork sneq:xing
.
```

```
######
## actor related properties
######

sneq:hasName a rdf:Property
; rdfs:label "hasName"
; rdfs:comment "Each person or group has a name"
; rdfs:domain sneq:Actor
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasWebsite a rdf:Property
; rdfs:label "hasWebsite"
; rdfs:comment "person or group can have a website"
; rdfs:domain sneq:Actor
; rdfs:range sneq:Website
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasDescription a rdf:Property
; rdfs:label "hasDescription"
; rdfs:comment "self info or description of the group"
; rdfs:domain sneq:Actor
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasNetwork a rdf:Property
; rdfs:label "hasNetwork"
; rdfs:comment "belong to a group"
; rdfs:domain sneq:Actor
; rdfs:range sneq:Network
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.


######
## person related properties to other classes
######

sneq:hasFriend a rdf:Property
; rdfs:label "hasFriend"
; rdfs:comment "persons friendship connection with another person"
; rdfs:domain sneq:Person
; rdfs:range sneq:Person
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.
```

```
sneq:hasWebProfile a rdf:Property
; rdfs:label "hasWebProfile"
; rdfs:comment "person other profiles on the web"
; rdfs:domain sneq:Person
; rdfs:range sneq:Website
; sneq:usedByNetwork sneq:xing
.

sneq:hasLection a rdf:Property
; rdfs:label "hasLection"
; rdfs:comment "person can have many lection"
; rdfs:domain sneq:Person
; rdfs:range sneq:Lection
; sneq:usedByNetwork sneq:studivz
.


######
## person related properties personal infos
######


sneq:hasJoinDate a rdf:Property
; rdfs:label "hasJoinDate"
; rdfs:comment "person join time to this social network"
; rdfs:domain sneq:Person
; rdfs:range xsd:dateTime
; sneq:usedByNetwork sneq:studivz
.


sneq:hasLastUpdate a rdf:Property
; rdfs:label "hasLastUpdate"
; rdfs:comment "persons last update"
; rdfs:domain sneq:Person
; rdfs:range xsd:dateTime
; sneq:usedByNetwork sneq:studivz
.


sneq:hasSex a rdf:Property
; rdfs:label "hasSex"
; rdfs:comment "sex of a person"
; rdfs:domain sneq:Person
; rdfs:range sneq:Sex
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.
```

```
sneq:hasInterestedIn a rdf:Property
; rdfs:label "hasInterestedIn"
; rdfs:comment "interested in male and/or woman"
; rdfs:domain sneq:Person
; rdfs:range sneq:Sex
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.

sneq:hasRelationshipStatus a rdf:Property
; rdfs:label "hasRelationshipStatus"
; rdfs:comment "whether person in relationship or not"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.

sneq:hasLookingFor a rdf:Property
; rdfs:label "hasLookingFor"
; rdfs:comment "persons goal why the person joined
    this social network"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasBirthday a rdf:Property
; rdfs:label "hasBirthday"
; rdfs:comment "birthday date of the person"
; rdfs:domain sneq:Person
; rdfs:range xsd:date
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasHometown a rdf:Property
; rdfs:label "hasHometown"
; rdfs:comment "persons hometown"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.

sneq:hasPoliticalView a rdf:Property
; rdfs:label "hasPoliticalView"
; rdfs:comment "persons political view"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.
```

```
sneq:hasReligiousView a rdf:Property
; rdfs:label "hasReligiousView"
; rdfs:comment "persons religious view"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:facebook
.

sneq:hasSkills a rdf:Property
; rdfs:label "hasSkills"
; rdfs:comment "persons skills"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.

sneq:hasWants a rdf:Property
; rdfs:label "hasWants"
; rdfs:comment "what person search"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.

sneq:hasHaves a rdf:Property
; rdfs:label "hasWants"
; rdfs:comment "what person offers"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.

sneq:hasOrganization a rdf:Property
; rdfs:label "hasOrganizaation"
; rdfs:comment "organization person is belonging"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.

sneq:hasHomeland a rdf:Property
; rdfs:label "hasHomeland"
; rdfs:comment "persons homeland"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz
.
```

```
######
## person related properties contact details
######

sneq:hasEmail a rdf:Property
; rdfs:label "hasEmail"
; rdfs:comment "persons email address"
; rdfs:domain sneq:Actor
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasIMName a rdf:Property
; rdfs:label "hasEmail"
; rdfs:comment "persons instant messaging nicknames"
; rdfs:domain sneq:Person
; rdfs:range sneq:IMName
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


sneq:belongsTo a rdf:Property
; rdfs:label "belongsTo"
; rdfs:comment "instant messaging nickname belongs to
    instant messaging service"
; rdfs:domain sneq:IMName
; rdfs:range sneq:IMService
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


sneq:hasMobile a rdf:Property
; rdfs:label "hasMobile"
; rdfs:comment "persons mobile number"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


sneq:hasLandPhone a rdf:Property
; rdfs:label "hasLandPhone"
; rdfs:comment "persons land phone number"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.
```

```
sneq:hasAddress a rdf:Property
; rdfs:label "hasAddress"
; rdfs:comment "persons address details"
; rdfs:domain sneq:Actor
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasBusinessMobile a rdf:Property
; rdfs:label "hasBusinessMobile"
; rdfs:comment "persons mobile number"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.

sneq:hasBusinessLandPhone a rdf:Property
; rdfs:label "hasBusinessLandPhone"
; rdfs:comment "persons land phone number"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.

sneq:hasBusinessAddress a rdf:Property
; rdfs:label "hasBusinessAddress"
; rdfs:comment "persons address details"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.

sneq:hasBusinessFax a rdf:Property
; rdfs:label "hasBusinessFax"
; rdfs:comment "persons fax contact number"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.

sneq:hasBusinessEmail a rdf:Property
; rdfs:label "hasBusinessEmail"
; rdfs:comment "persons email address"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.
```

```
######
## person related properties liking
######


sneq:hasActivities a rdf:Property
; rdfs:label "hasActivities"
; rdfs:comment "persons activities"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.

sneq:hasInterests a rdf:Property
; rdfs:label "hasInterests"
; rdfs:comment "persons interests"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasFavMusik a rdf:Property
; rdfs:label "hasFavMusik"
; rdfs:comment "persons favorite musik"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.


sneq:hasFavTvShow a rdf:Property
; rdfs:label "hasFavTvShow"
; rdfs:comment "persons favorite tv show"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:facebook
.


sneq:hasFavMovie a rdf:Property
; rdfs:label "hasFavMovie"
; rdfs:comment "persons favorite movie"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.
```

```
sneq:hasFavBook a rdf:Property
; rdfs:label "hasFavBook"
; rdfs:comment "persons favorite book"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.

sneq:hasFavQuote a rdf:Property
; rdfs:label "hasFavQuote"
; rdfs:comment "persons favorite quote"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.


######
## person related properties job
######


sneq:hasJob a rdfs:Class
; rdfs:label "hasJob"
; rdfs:comment "property for blank node job"
; rdfs:domain sneq:Person
; rdfs:range sneq:Job
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


sneq:hasEmployer a rdf:Property
; rdfs:label "hasEmployer"
; rdfs:comment "where or for whom person work"
; rdfs:domain sneq:Job
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


sneq:hasPosition a rdf:Property
; rdfs:label "hasPosition"
; rdfs:comment "persons position at work"
; rdfs:domain sneq:Job
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.
```

```
sneq:hasWorkLocation a rdf:Property
; rdfs:label "hasWorkLocation"
; rdfs:comment "where person works"
; rdfs:domain sneq:Job
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasJobDescription a rdf:Property
; rdfs:label "hasDescription"
; rdfs:comment "Desription of the kind of work person do"
; rdfs:domain sneq:Job
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasIndustry a rdf:Property
; rdfs:label "hasIndustry"
; rdfs:comment "industry of company"
; rdfs:domain sneq:Job
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.

sneq:hasAdditionalIndustry a rdf:Property
; rdfs:label "hasAdditionalIndustry"
; rdfs:comment "other industries of company"
; rdfs:domain sneq:Job
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.

sneq:hasJobStatus a rdf:Property
; rdfs:label "hasJobStatus"
; rdfs:comment "industry of company"
; rdfs:domain sneq:Job
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.


sneq:hasCareerLevel a rdf:Property
; rdfs:label "hasCareerLevel"
; rdfs:comment "career level of the person in the company"
; rdfs:domain sneq:Job
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.
```

```
sneq:hasCompanySize a rdf:Property
; rdfs:label "hasCompanySize"
; rdfs:comment "company size"
; rdfs:domain sneq:Job
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.

sneq:hasOrganizationType a rdf:Property
; rdfs:label "hasOrganizationType"
; rdfs:comment "organization type"
; rdfs:domain sneq:Job
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.


sneq:hasCompanySite a rdf:Property
; rdfs:label "hasCompanySize"
; rdfs:comment "companies website"
; rdfs:domain sneq:Job
; rdfs:range sneq:Website
; sneq:usedByNetwork sneq:xing
.


######
## person related properties job experience
######


sneq:hasExperience a rdf:Property
; rdfs:label "hasExperience"
; rdfs:comment "persons professional experience"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


sneq:hasAwards a rdf:Property
; rdfs:label "hasAwards"
; rdfs:comment "persons professional achievments"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.
```

```
sneq:hasStatus a rdf:Property
; rdfs:label "hasStatus"
; rdfs:comment "persons professional status"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing
.


######
## person related properties education
######


sneq:hasEducation a rdf:Property
; rdfs:label "hasEducation"
; rdfs:comment "property for blank node education"
; rdfs:domain sneq:Person
; rdfs:range sneq:Education
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasCollege a rdf:Property
; rdfs:label "hasCollege"
; rdfs:comment "persons college"
; rdfs:domain sneq:Education
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.


sneq:hasHighScool a rdf:Property
; rdfs:label "hasHighScool"
; rdfs:comment "persons highscool, facebook"
; rdfs:domain sneq:Education
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:facebook
.


sneq:hasEducationName a rdf:Property
; rdfs:label "hasEducationName"
; rdfs:comment "name for education"
; rdfs:domain sneq:Education
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.
```

```
sneq:hasEducationStartDate a rdf:Property
; rdfs:label "hasEducationStartDate"
; rdfs:comment "education started"
; rdfs:domain sneq:Education
; rdfs:range xsd:date
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasEducationEndDate a rdf:Property
; rdfs:label "hasEducationEndDate"
; rdfs:comment "education ended"
; rdfs:domain sneq:Education
; rdfs:range xsd:date
; sneq:usedByNetwork sneq:xing, sneq:studivz, sneq:facebook
.

sneq:hasLanguage a rdf:Property
; rdfs:label "hasLanguage"
; rdfs:comment "language person talk"
; rdfs:domain sneq:Person
; rdfs:range rdfs:Literal
; sneq:usedByNetwork sneq:studivz, sneq:facebook
.
```

B Beispiel für extrahierte Profildaten

```
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:    <http://xmlns.com/foaf/0.1/> .


<http://www.xam.de/2007/sneq#hasSocialNetwork:facebook>
      a       <http://www.xam.de/2007/sneq#SocialNetwork> .

<http://www.xam.de/2007/sneq#hasSocialNetwork:studivz>
      a       <http://www.xam.de/2007/sneq#SocialNetwork> .

<http://www.xam.de/2007/sneq#hasSocialNetwork:xing>
      a       <http://www.xam.de/2007/sneq#SocialNetwork> .


<http://www.xam.de/2007/sneq#hasSocialNetwork:sexFemale>
      a       <http://www.xam.de/2007/sneq#Sex> .


<http://www.facebook.com/profile.php?id=191567972>
      a       <http://www.xam.de/2007/sneq#Person> ;
      rdfs:label "191567972" ;
      <http://www.xam.de/2007/sneq#hasBirthday>
              "1981-05-28T00:00:00Z"^^xsd:dateTime ;
      <http://www.xam.de/2007/sneq#hasFriend>
              <http://www.facebook.com/profile.php?id=128733889> ,
              <http://www.facebook.com/profile.php?id=275054863> ,
              <http://www.facebook.com/profile.php?id=107602757> ;
      <http://www.xam.de/2007/sneq#hasLookingFor>
              "Friendship" ;
      <http://www.xam.de/2007/sneq#hasName>
              "Kathrin Hartmann" ;
      <http://www.xam.de/2007/sneq#hasSocialNetwork>
              <http://www.xam.de/2007/sneq#hasSocialNetwork
                                                  :facebook> .


<http://www.facebook.com/profile.php?id=128733889>
      a       <http://www.xam.de/2007/sneq#Person> ;
      rdfs:label "128733889" ;
      <http://www.xam.de/2007/sneq#hasName>
              "Nicole Wagner" ;
      <http://www.xam.de/2007/sneq#hasSocialNetwork>
              <http://www.xam.de/2007/sneq#hasSocialNetwork
                                                  :facebook> .
```

```
<http://www.facebook.com/profile.php?id=275054863>
      a       <http://www.xam.de/2007/sneq#Person> ;
      rdfs:label "275054863" ;
      <http://www.xam.de/2007/sneq#hasName>
              "Elizabeth Hirsch" ;
      <http://www.xam.de/2007/sneq#hasSocialNetwork>
              <http://www.xam.de/2007/sneq#hasSocialNetwork
                                                  :facebook> .

<http://www.facebook.com/profile.php?id=107602757>
      a       <http://www.xam.de/2007/sneq#Person> ;
      rdfs:label "107602757" ;
      <http://www.xam.de/2007/sneq#hasName>
              "Martin Fleischmann" ;
      <http://www.xam.de/2007/sneq#hasSocialNetwork>
              <http://www.xam.de/2007/sneq#hasSocialNetwork
                                                   :facebook> .

<http://www.studivz.net/Profile/fd73e9be330ae860>
      a       <http://www.xam.de/2007/sneq#Person> ;
      rdfs:label "fd73e9be330ae860" ;
      <http://www.xam.de/2007/sneq#hasBirthday>
              "1981-05-28T00:00:00Z"^^xsd:dateTime ;
      <http://www.xam.de/2007/sneq#hasEducation>
              <urn:rnd:-20afe65c:119239191dc:-7ffd> ;
      <http://www.xam.de/2007/sneq#hasFriend>
              <http://www.studivz.net/Profile/702cc8f13bcef006> ;
      <http://www.xam.de/2007/sneq#hasName>
              "Kathrin Hartmann" ;
      <http://www.xam.de/2007/sneq#hasSex>
              <http://www.xam.de/2007/sneq#hasSocialNetwork
                                                  :sexFemale> ;
      <http://www.xam.de/2007/sneq#hasSocialNetwork>
              <http://www.xam.de/2007/sneq#hasSocialNetwork
                                                  :studivz> .

<urn:rnd:-20afe65c:119239191dc:-7ffd>
      a       <http://www.xam.de/2007/sneq#Education> ;
      <http://www.xam.de/2007/sneq#hasCollege>
              "FH Fulda" .

<http://www.studivz.net/Profile/702cc8f13bcef006>
      a       <http://www.xam.de/2007/sneq#Person> ;
      rdfs:label "702cc8f13bcef006" ;
      <http://www.xam.de/2007/sneq#hasName>
              "Karsten Klein" ;
      <http://www.xam.de/2007/sneq#hasSocialNetwork>
              <http://www.xam.de/2007/sneq#hasSocialNetwork
                                                   :studivz> .
```

```
<https://www.xing.com/profile/Kathrin_Hartmann>
      a       <http://www.xam.de/2007/sneq#Person> ;
      rdfs:label "Kathrin_Hartmann" ;
      <http://www.xam.de/2007/sneq#hasBusinessAddress>
              "81379 München" ;
      <http://www.xam.de/2007/sneq#hasEducation>
              <urn:rnd:-20afe65c:119239191dc:-7ffa> ;
      <http://www.xam.de/2007/sneq#hasFriend>
              <https://www.xing.com/profile/Elizabeth_Hirsch> ,
              <https://www.xing.com/profile/Alexander_Schubert> ;
      <http://www.xam.de/2007/sneq#hasInterests>
              "philosophy" ,
              "personal knowledge management" ,
               "art" ;
      <http://www.xam.de/2007/sneq#hasJob>
              <urn:rnd:-20afe65c:119239191dc:-7ffb> ;
      <http://www.xam.de/2007/sneq#hasLanguage>
              "German, English" ;
      <http://www.xam.de/2007/sneq#hasName>
              "Kathrin Hartmann" ;
      <http://www.xam.de/2007/sneq#hasSkills>
              "Product management" ,
              "moderation" ,
              "usability" ,
              "design and information and communication software" ,
              "Experience in web development" ;
      <http://www.xam.de/2007/sneq#hasSocialNetwork>
              <http://www.xam.de/2007/sneq#hasSocialNetwork:xing> .

<urn:rnd:-20afe65c:119239191dc:-7ffa>
      a       <http://www.xam.de/2007/sneq#Education> ;
      <http://www.xam.de/2007/sneq#hasCollege>
              "FH Fulda" .

<urn:rnd:-20afe65c:119239191dc:-7ffb>
      a       <http://www.xam.de/2007/sneq#Job> ;
      <http://www.xam.de/2007/sneq#hasEmployer>
              "Deutsches Rotes Kreuz" .

<https://www.xing.com/profile/Elizabeth_Hirsch>
      a       <http://www.xam.de/2007/sneq#Person> ;
      rdfs:label "Elizabeth_Hirsch" ;
      <http://www.xam.de/2007/sneq#hasName>
              "Elizabeth Hirsch" ;
      <http://www.xam.de/2007/sneq#hasSocialNetwork>
              <http://www.xam.de/2007/sneq#hasSocialNetwork:xing> .
```

```
<https://www.xing.com/profile/Alexander_Schubert>
      a       <http://www.xam.de/2007/sneq#Person> ;
      rdfs:label "Alexander_Schubert" ;
      <http://www.xam.de/2007/sneq#hasName>
              "Alexander Schubert" ;
      <http://www.xam.de/2007/sneq#hasSocialNetwork>
              <http://www.xam.de/2007/sneq#hasSocialNetwork:xing> .
```

Literatur

[BBL08] David Beckett and Tim Berners-Lee. Turtle - terse rdf triple language. W3C Team Submission, W3C, January 2008.

[BL98] Berners-Lee. Notation 3, 1998.

[Bou92] Pierre Bourdieu. *Die verborgenen Mechanismen der Macht.*, pages 49–80. Vsa Verlag, 1992.

[BPSM+04] Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, Eve Maler, François Yergeau, and John Cowan. Extensible Markup Language (XML) 1.1. Recommendation, World Wide Web Consortium (W3C), February 2004. http://www.w3.org/TR/2004/REC-xml11-20040204/.

[CD99] James Clark and Steve DeRose. XML Path Language (XPath) Version 1.0. Technical report, W3C, November 1999.

[DB04a] Libby Miller Dan Brickley. FOAF Vocabulary Specification, April 2004.

[DB04b] R.V. Guha Dan Brickley. RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/, February 2004.

[DS04] Mike Dean and Guus Schreiber. OWL Web Ontology Language - Reference. Recommendation, W3C, February 2004.

[ea02] Steven Pemberton et al. XHTML™1.0 the extensible hypertext markup language (second edition) – a reformulation of HTML 4 in XML 1.0. Technical report, W3C, 2002. W3C Recommendation 26 January 2000, revised 1 August 2002.

[EIV07] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *Knowledge and Data Engineering, IEEE Transactions on*, 19(1):1–16, 2007.

[ES] M. Ehrig and Y. Sure. Ontology mapping — an integrated approach.

[GHM+07] Tudor Groza, Siegfried Handschuh, Knud Moeller, Gunnar Grimnes, Leo Sauermann, Enrico Minack, Cedric Mesnage, Mehdi Jazayeri, Gerald Reif, and Rosa Gudjonsdottir. The nepomuk project - on the way to the social semantic desktop. In Tassilo Pellegrini and Sebastian Schaffert, editors, *Proceedings of I-Semantics' 07*, pages pp. 201–211. JUCS, 2007.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[KC04] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, February 2004.

[Kur06] Andreas Kurz. Integration von Wiki-Inhalten. Studienarbeit, Universität Karlsruhe, 12 2006.

[Lev66] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

[LF99] Berners T. Lee and M. Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. Harper San Francisco, 1999.

[PS08] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. Recommentation, W3C, January 2008.

[Völ03] Max Völkel. Extraktion von XML aus HTML-Seiten. Diplomarbeit, Universität Karlsruhe, 05 2003.

[Völ06] Max Völkel. RDFReactor – From Ontologies to Programatic Data Access. In *Proc. of the Jena User Conference 2006*. HP Bristol, MAY 2006.

[WFI94] Stanley Wasserman, Katherine Faust, and Dawn Iacobucci. *Social Network Analysis : Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, November 1994.