

Friedrich-Schiller-Universität Jena

Philosophische Fakultät

Institut für Germanistische Sprachwissenschaft

Retrieval und Visualisierung biomedizinischer Relationen

Erweiterung einer semantischen Suchmaschine zur Arbeit mit

Protein-Protein-Interaktionen

Masterarbeit zur Erlangung des akademischen Grades

Master of Arts (M.A.)

vorgelegt von Johannes Hellrich

geboren am 24.9.1982 in Schweinfurt

Erstgutachter: Prof. Dr. Udo Hahn

Zweitgutachterin: HDoz. Dr. Christine Römer

Jena, 28. März 2012

Abstract

Moderne Information Retrieval Systeme helfen Forschern bei der Suche nach den für sie relevanten Publikationen. Diese Arbeit beschreibt die Erweiterung SEMEDICOS, einer biomedizinischen Suchmaschine, zur Arbeit mit Protein-Protein-Interaktionen. Dazu wurde ein Prototyp entwickelt, der die Interaktionsdaten speichern, durchsuchen und als Graph visualisieren kann. Speicherung und Suche basieren auf RDF, der vom W3C standardisierten Basis des Semantic Webs. Die interaktiven Graphen werden mit CYTOSCAPE WEB erzeugt, der online Version der populärsten biomedizinischen Visualisierungssoftware. Durch diese Kombination aus Suchmaschine und Visualisierung ergibt sich ein neuartiges Hilfsmittel für die biomedizinische Forschung.

Modern information retrieval systems help scientists searching for relevant publications. This thesis describes improvements to the biomedical search engine SEMEDICO to enable storing, retrieval and visualization of protein-protein interactions. A prototype was developed, which uses the W3C standard RDF to store and retrieve the interactions. CYTOSCAPE WEB, an online version of the most popular biomedical visualization tool, is used to display interactive interaction networks. This combination, out of a search engine and an online visualization tool, would be a new kind of tool for biomedical researchers.

Inhaltsverzeichnis

1	Informationsflut in der Biomedizin	1
2	Biomedizin und Computerlinguistik	5
2.1	Wichtige Konzepte	5
2.2	Biomedizinische Datenbanken	8
2.3	Biomedizinische Suchmaschinen	10
2.4	SEMEDICO	12
2.5	JREX	16
2.6	Visualisierungssoftware	18
3	Technologien im Detail	24
3.1	RDF	24
3.1.1	Aufbau und Anfragen	25
3.1.2	Einsatz	29
3.2	CYTOSCAPE WEB	32
4	Implementierung	36
4.1	Speicherung	37
4.2	Zugriff auf die Relationen	40
4.2.1	Parser und SOLR-Query	41
4.2.2	SPARQL in SEMEDICO	44
4.3	Anzeigen der Ergebnisse	47
5	Zusammenfassung und Ausblick	53

Kapitel 1

Informationsflut in der Biomedizin

Die Geschwindigkeit, mit der sich Kultur und Wissenschaft entwickeln, macht es einem Einzelnen schon lange unmöglich, alles neu Publierte zu lesen [Michel u. a. 2011; Hersh 2003, Kapitel 2]. Ein Gebiet, in dem diese Informationsflut immer mehr zur Herausforderung wird, ist die Biomedizin, der Übergangsbereich zwischen medizinischer und biologischer Grundlagenforschung. Abbildung 1.1 zeigt das exponentielle Wachstum der biomedizinischen Veröffentlichungen. Die Abbildung basiert auf den Einträgen in MEDLINE, einer seit 1946 von der U.S. Nationalbibliothek betriebenen Sammlung biomedizinischer Artikel.¹

Wissenschaftler und Ärzte verbringen, wegen dieser Informationsflut, immer mehr Zeit mit Recherchen, erhalten aber keine befriedigenden Ergebnisse [Schneider 2007]. Zur Lösung dieses Problems wird von vielen Experten der Einsatz computerlinguistischer Methoden vorgeschlagen, um Texte besser zu erschließen [Altman u. a. 2008]. Computerlinguistische Systeme können ihre Benutzer auf verschiedene Arten unterstützen, wie etwa durch den Abgleich von Krankenakten mit Medizinischen Leitlinien [Milian u. a. 2009]. Dabei sollten sie in Teilbereichen, wie etwa der automatischen Verschlagwortung biomedizinischer Veröffentlichungen, nicht nur schneller, sondern auch ge-

¹www.nlm.nih.gov/pubs/factsheets/medline.html

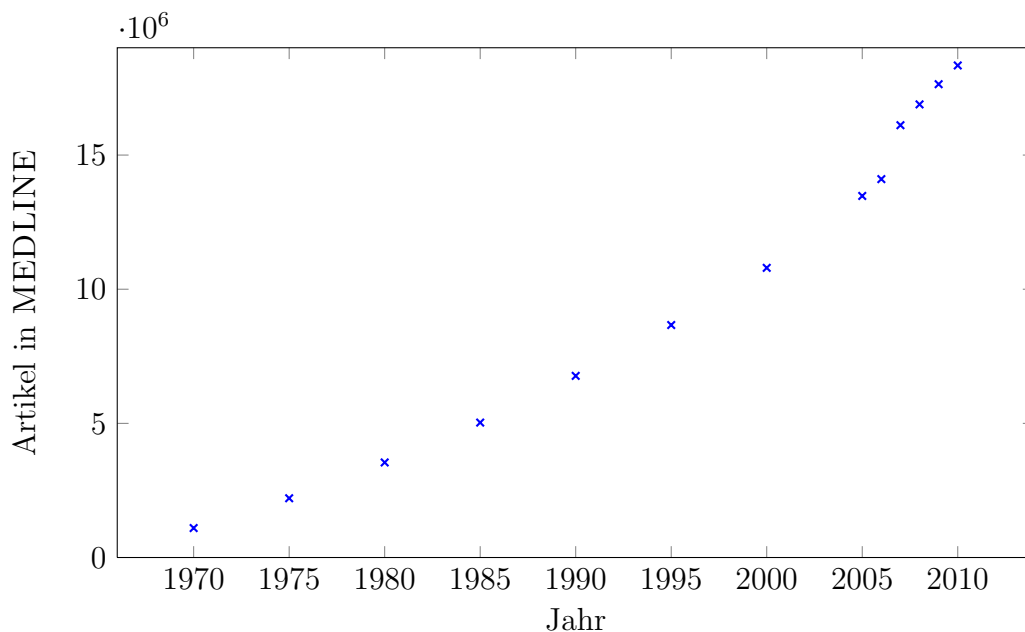


Abbildung 1.1: MEDLINE Einträge für 1970–2010.

nauer als Menschen arbeiten [Hahn u. a. 2007a].

Diese Arbeit behandelt den Einsatz computerlinguistischer Systeme zur Entwicklung semantischer Suchmaschinen. Diese finden Texte nicht über die darin enthaltenen Wörter, sondern über die Bedeutung dieser Wörter. Dafür sind sowohl Information Extraction 'Informationsextraktion' (IE), als auch Information Retrievals 'Informationsrückgewinnung' (IR) nötig – also die Untersuchung der Texte darauf, wofür sie relevant sind und die Suche nach Texten, die für eine Anfrage relevant sind. Dabei gibt es unterschiedlich anspruchsvolle Ansätze zur Erschließung. Im einfachsten Fall werden lediglich Schreibvarianten normalisiert. Wesentlich anspruchsvoller ist die Untersuchung des Textinhalts auf die Zusammenhänge zwischen den darin beschriebenen Entitäten [Friedman u. Johnson 2006]. Diese Zusammenhänge sind für die biomedizinische Forschung von großer Bedeutung. Die biomedizinische Forschung fokussiert sich, nach der Entschlüsselung des menschlichen Genoms, auf die Interaktionen der darin kodierten Proteine [Stumpf u. a. 2008; Ge u. a. 2003].

Im Folgenden wird beschrieben, wie die semantische Suchmaschine, SEME-

DICO², erweitert wurde, um die Speicherung, Abfrage und Visualisierung biomedizinischer Interaktionen zu ermöglichen. Ein entsprechender Prototyp wurde im Rahmen der Masterarbeit entwickelt. Die Visualisierung der Interaktionsnetzwerke ist nötig, da sich Forscher nur so einen Überblick darüber verschaffen können [Fluit u. a. 2002; Searls 2005].

Diese Kombination aus Suchmaschine und Visualisierung ist ein neuartiges Hilfsmittel für die biomedizinische Forschung. Im nächsten Kapitel werden die theoretischen Grundlagen und der Forschungsstand biomedizinischer IR- und Visualisierungssysteme beschrieben. Kapitel 3 beschreibt die zur Speicherung und Visualisierung der Interaktionen genutzten Systeme genauer. Kapitel 4 erläutert, wie diese zur Implementierung des Prototypen einer nächsten SEMEDICO Version genutzt wurden. Kapitel 5 fasst die Ergebnisse zusammen und macht Vorschläge für zukünftige Arbeiten.

²www.semedico.org/app

Kapitel 2

Biomedizin und Computerlinguistik

Dieses Kapitel behandelt den allgemeinen Stand und die theoretischen Hintergründe der biomedizinischen IR Forschung und der Visualisierungssysteme.

2.1 Wichtige Konzepte

Im Folgenden werden einige für das Verständnis der Arbeit grundlegende Konzepte eingeführt. Dabei werden mit Graphen und Ontologien zwei in der Informatik weit verbreitete Konstrukte skizziert, die Terme *Information Retrieval* und *Information Extraction* erläutert und das Verhältnis der Computerlinguistik zur Biomedizin angesprochen.

Graphen sind als das Paar aus der Menge der Knoten und der Menge ihrer Verbindungen, Kanten genannt, definiert [Ottmann u. Widmayer 2002, Kapitel 8; Cormen u. a. 2001, Appendix B]. Dies wird in Abbildung 2.1 verdeutlicht. In der Linguistik sind (Syntax-)Bäume als Spezialfall der Graphen weit verbreitet. Graphen werden in beinahe jedem der folgenden Schritte genutzt, sie können zur Speicherung, Verarbeitung und Anzeige von Informationen eingesetzt werden.

Ontologien sind zwischen Systemen übertragbare Modelle für das menschl-

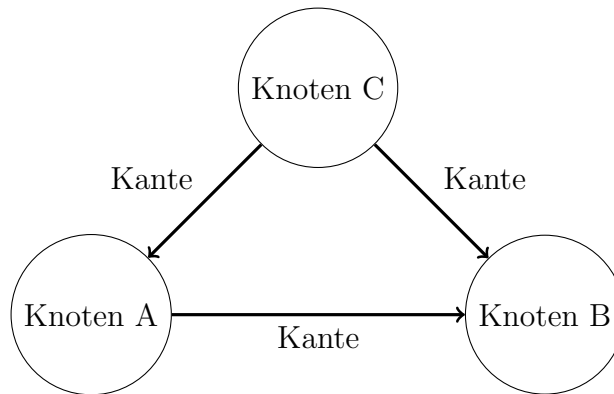


Abbildung 2.1: Beispiel für einen Graphen.
 $G = \langle V, E \rangle$, $V = \{A, B, C\}$, $E = \{\langle A, B \rangle, \langle C, B \rangle, \langle C, A \rangle\}$

che Weltwissen [Vrandecic 2010; Uschold u. Gruninger 2004]. Sie werden, um Ambiguität zu vermeiden, in formalen Sprachen verfasst und decken ihren Themenbereich möglichst komplett ab. Ontologien bestehen aus Begriffen und deren Relationen. Ein Beispiel sind die Begriffe *Philosoph* und *Person*, zwischen denen die Relation *is-a*(x, y) besteht – jeder *Philosoph* ist eine *Person*, aber nicht umgekehrt. Begriffe können instanziiert werden, so ist etwa *Sokrates* eine Instanz von *Philosoph*. Relationen gelten nicht nur zwischen Begriffen, sondern auch zwischen ihren Instanzen. Ontologien sind geeignet, um automatisch die Wahrheit logischer Aussagen, über den von ihnen behandelten Bereich, zu beurteilen und neue Schlüsse zu ziehen. Dies ist mit den so genannten Ontologien der in diesem Kapitel behandelten Systeme nicht oder nur eingeschränkt möglich. Sie müssten, je nach Grad ihrer Formalisierung, als Thesaurus oder Taxonomie bezeichnet werden, da sie sich lediglich mit Synonymie und Hyponymie/Hyperonymie beschäftigen. Ich werde, nach diesem Hinweis, der verbreiteten Praxis und den Selbstbezeichnungen folgen und Ontologie ab jetzt als Oberbegriff für Thesauri, Taxonomien und die eigentlichen Ontologien nutzen.

Um die Inhalte einer Publikation über eine Suchmaschine zugänglich zu machen, sind Information Retrieval (IR) und Information Extraction (IE) nötig. Beim Information Retrieval geht es darum, zu einer Suchanfrage die Dokumente zu finden, die dieser am besten entsprechen [Hersh 2003; Jurafsky u. Martin 2009, Kapitel 23]. Dabei bezeichnet *Dokument* beim IR einen be-

liebigen Datenbankeintrag, etwa ein Buch oder ein Zeitschriftenartikel. Ein typisches Beispiel sind Suchmaschinen wie Google¹, die zur Anfrage passende Webseiten als Antwort liefern. Die technische Herausforderung besteht hier in der uneinheitlichen Struktur der Einträge, anders als bei einer Relationalen Datenbank gibt es keine eindeutigen Schlüssel, die das System nutzen könnte. Dagegen ist Information Extraction die Suche nach Informationen im Inhalt eines Texts [Jurafsky u. Martin 2009, Kapitel 22]. Diese können dann etwa genutzt werden, um den Text in einem IR System besser zu indizieren. Bei der IE können semantische Informationen mit unterschiedlicher Komplexität aus dem Text extrahiert werden. Im einfachsten Fall werden Entitäten, wie etwa Personen, Orte oder Proteine identifiziert. Komplexere Anwendungen können die Relationen zwischen diesen Entitäten extrahieren, wie etwa biomedizinisch relevante Abläufe.

In der IE-Forschung entwickelte sich, mit der vom amerikanischen Militär geförderten Message Understanding Conference (MUC) und ihren Nachfolgern, ein neuartiges, wettbewerbsorientiertes Konferenzformat, die so genannten Shared Tasks [Ralph Grishman 1996]. Bei diesen vergleichen die Teilnehmer ihre IE-Systeme anhand eines vom Konferenzveranstalter erstellten Korpus. Dieses muss annotiert werden, eine Musterlösung liegt vor, steht den Teilnehmern zur Entwicklung ihres Systems aber nur in Auszügen zur Verfügung. Für den Anwendungsbereich biomedizinische Texte sind etwa die BIONLP Shared Tasks² und BIOCREATIVE³ relevant. In der BIONLP'09⁴ mussten die Teilnehmer Events unter Beteiligung von Proteinen oder Genen⁵ extrahieren. Diese Protein-Protein-Interaktionen (PPI) wurden in den letzten Jahren zu einem zentralen Thema der biomedizinischen Forschung [Stumpf u. a. 2008; De Las Rivas u. Fontanillo 2010]. PPIs regeln die Abläufe in Organismen, ihre Anzahl korreliert stärker mit der Komplexität eines Organismus als die Größe seines Genoms. So werden etwa für den Menschen rund 650.000 PPIs erwartet, für *S. cerevisiae* (Bäckerhefe) dagegen nur ca.

¹www.google.com

²sites.google.com/site/bionlpst/

³www.biocreative.sourceforge.net

⁴www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/

⁵Da Gene für Proteine kodieren, werden beide Bezeichnungen oft austauschbar gebraucht.

35.000. Die Erkennung der PPIs in wissenschaftlichen Publikationen wird durch die unterschiedlichen Bezeichnungen für Proteine und Interaktionen erschwert. So wird etwa das Protein *MET_HUMAN* in *the beta subunit regulated MAT II activity by reducing its K(m) for L-Met* durch *beta_subunit* repräsentiert.

2.2 Biomedizinische Datenbanken

Biomedizinische Datenbanken erleichtern es Forschern ihre Ergebnisse miteinander zu teilen. Die wohl wichtigste (Meta-)Datenbank für den biomedizinischen Bereich ist PUBMED⁶. PUBMED basiert auf MEDLINE und weist jedem Artikel eine eindeutige Nummer, seine PMID (PUBMED-ID) zu. Wie eingangs schon erwähnt und in Grafik 1.1 erkennbar, wächst der in PUBMED verzeichnete Datensatz exponentiell. PUBMED erlaubt die Suche anhand verschiedener Kriterien,⁷ zentral sind dabei die jedem Artikel von Hand zugewiesenen MeSH⁸ (Medical Subject Headings) Terme. Die Mehrzahl der Anfragen erfolgt über die Suchbox, die darin verwendeten Wörter werden über eine Tabelle auf die Terme des MeSH Vokabulars abgebildet, etwa *black_death* auf *plague* [Knecht u. Nelson 2002]. Besonders relevant für die Arbeit mit Proteinen ist UNIPROT⁹ (Universal Protein Resource) [Wu u. a. 2006]. UNIPROT ist eine Oberfläche zur Abfrage mehrerer thematisch eng verwandter Datenbanken und fungiert als zentrales Verzeichnis aller bekannten Proteine. Es speichert deren Aminosäuresequenzen, biologische Funktion und Verweise auf die relevante Literatur. Zur Erhöhung der Interoperabilität können Einträge anderer Datenbanken automatisiert auf ihre Gegenstücke in UNIPROT abgebildet werden.¹⁰ Speziellere Datenbanken verweisen regelmäßig darauf und nutzen es, um ihre Einträge zu verifizieren und zu kontextualisieren [Kerrien u. a. 2012].

⁶www.ncbi.nlm.nih.gov/pubmed

⁷www.ncbi.nlm.nih.gov/books/NBK3830/

⁸www.nlm.nih.gov/mesh/meshhome.html

⁹www.uniprot.org/

¹⁰www.uniprot.org/help/mapping

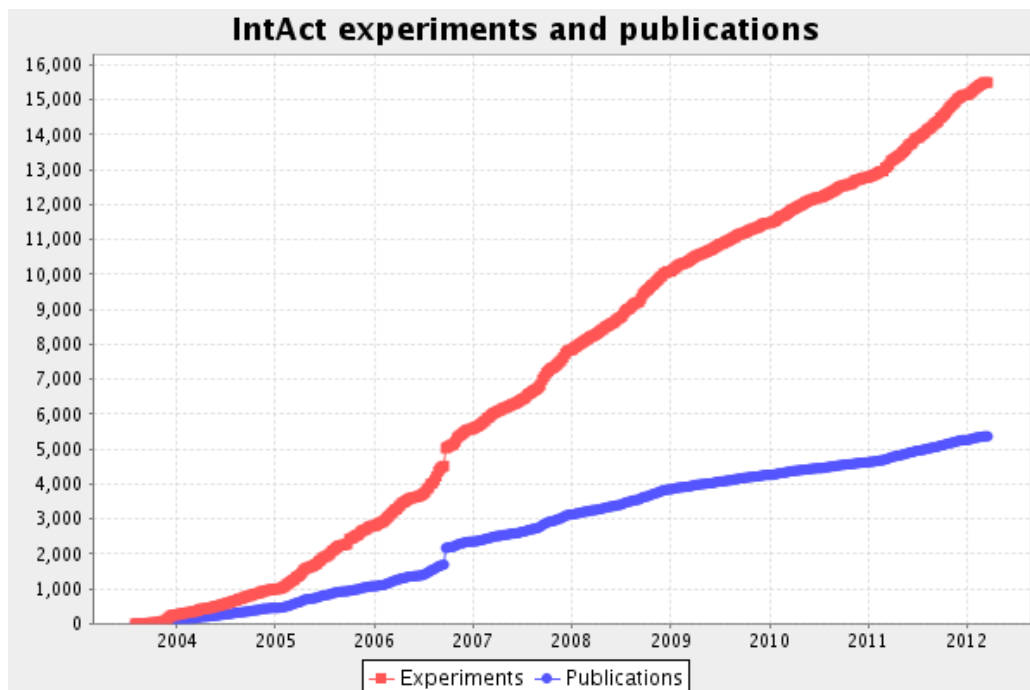


Abbildung 2.2: Artikel und Experimente in INTACT (Abgerufen am 15.2.2012 unter www.ebi.ac.uk/intact/statisticView/do/statistics).

Auch für PPIs gibt es speziellen Datenbanken, etwa MINT¹¹ (Molecular INteraction database), DIP¹² (Database of Interacting Proteins) und INTACT¹³ (INTACT molecular interaction database) [Ceol u. a. 2010; Xenarios u. a. 2000; Kerrien u. a. 2012]. Deren Einträge werden aktuell von Kuratoren erstellt, die dazu die Volltexte einiger ausgewählter Fachzeitschriften lesen. Die Kuratoren sind dabei nur für Übertragung und Strukturierung der Daten zuständig, nicht für deren Bewertung. So wird etwa für MINT ein Kurationsprozess mit zwei sich gegenseitig überwachenden Wissenschaftlern beschrieben, die aus ausgewählten Fachzeitschriften alle Proteininteraktionen in die Datenbank eintragen. Dabei wird auch das experimentelle Vorgehen vermerkt, der Benutzer kann dadurch selbst entscheiden, für wie aussagekräftig er die Ergebnisse hält. Durch den aufwändigen Kurationsprozess können derartige Datenbanken nur einen vergleichsweise geringen Teil der Literatur

¹¹mint.bio.uniroma2.it/mint

¹²dip.doe-mbi.ucla.edu/dip

¹³www.ebi.ac.uk/intact

abdecken, so hatte etwa INTACT im Herbst 2011 nur ca. fünftausend Artikel [Kerrien u. a. 2012]. Dabei wächst die Anzahl der verzeichneten Artikel linear, wie in Abbildung 2.2 erkennbar ist. Dies steht im Widerspruch zum exponentiellen Wachstum der biomedizinischen Literatur (vgl. Abbildung 1.1). Der beschriebene Kurationsprozess ist somit unzureichend, schnellere Methoden müssen gefunden werden, damit alle relevanten Veröffentlichungen erfasst werden können.

2.3 Biomedizinische Suchmaschinen

Neben den Datenbanken, die Rohdaten oder aufbereitete Informationen aus wissenschaftlichen Zeitschriften enthalten, gibt es auch spezialisierte Suchmaschinen, die auf relevante Publikationen verweisen. Es gibt mehrere Systeme, die versuchen PUBMEDs Suche zu ersetzen oder deren Ergebnisse weiter aufzubereiten. Hierfür wurden zwei grundsätzliche Verbesserungsmöglichkeiten vorgeschlagen [Lu 2011]: Einerseits kann die Suchanfrage besser auf die Inhalte der Datenbank abgebildet werden, andererseits können die Ergebnisse nach einem besseren System geordnet oder aufbereitet werden. Die meisten alternativen Suchmaschinen setzen bei der Präsentation der Ergebnisse an und bieten etwa interaktive Sortierungen oder graphische Anzeigen für verwandte Themen und Arbeiten. Im Folgenden werden einige ausgewählte Suchmaschinen genauer behandelt, weitere finden sich in einer betreuten Datenbank¹⁴. GOPUBMED¹⁵ bietet eine bessere Aufbereitung der Suchergebnisse als PUBMED, greift aber intern auf dieses zurück [Doms u. Schroeder 2005]. Aus den Abstracts der in PUBMED gefundenen Artikel werden hierzu Schlüsselwörter extrahiert, die mit der GO (Gene Ontology) abgeglichen werden. Dadurch können PUBMEDs Ergebnisse für den Benutzer hierarchisch und thematisch sortiert angezeigt werden. Man bezeichnet dies als Facettierung, eine etwa im E-Commerce weit verbreitete Lösung.¹⁶

¹⁴www.ncbi.nlm.nih.gov/CBBresearch/Lu/search/

¹⁵www.gopubmed.org

¹⁶vgl. die linken Seitenleisten auf www.amazon.de/ oder www.ebay.de/

Einen ähnlichen Ansatz verfolgt TEXTPRESSO¹⁷, wobei hier nicht die GO, sondern eine vergleichsweise kleine und flache Ontologie zur Anwendung kommt [Müller u. a. 2003]. Es gibt TEXTPRESSO Varianten für verschiedene Organismen, ursprünglich wurde das System für Artikel über den Fadenwurm *Caenorhabditis elegans* entwickelt. Das System nutzt reguläre Ausdrücke, insgesamt 14.500, zur Erkennung der Terme. Reguläre Ausdrücke passen auf mehrere Zeichenketten, so kann man etwa mit *[sS]imultaneous(ly)?* nach *simultaneous*, *simultaneously* und deren großgeschriebenen Varianten suchen [Jurafsky u. Martin 2009, Kapitel 2].

Ein Beispiel für ein System, das schon bei der Anfrage ansetzt, ist das inzwischen eingestellte TAMBIS¹⁸ (Transparent Access to Multiple Bioinformatics Information Sources) [Stevens u. a. 2000]. TAMBIS nutzt Ontologien, um die Nutzeranfragen auf verschiedene spezialisierte Datenbanken zu verteilen.

Eines der am weitesten entwickelten Alternativsysteme ist FACTA+¹⁹ [Tsuruoka u. a. 2011, 2008]. FACTA+ sucht nicht nur nach Wörtern, sondern auch nach Termen, für die ein eigener Invertierter Index (vgl. Seite 15) angelegt ist. Da diese semantischen Informationen bereits vor der Abfrage extrahiert wurden, kann das System Anfragen schnell und flexibel beantworten. Die Erkennung der Terme erfolgt mit Hilfe mehrerer Datenbanken und Ontologien, durch die verschiedene Repräsentationen eines Terms im Text gefunden werden können. Die Ergebnisse einer Suchanfrage werden in mehreren Tabellen präsentiert, die allgemeinen Termen wie *Disease* (engl. 'Krankheit') entsprechen. In diesen Tabellen werden weitere Terme angezeigt, die zusammen mit dem gesuchten Term in den Texten gefunden wurden. Sie sind nach ihrer Häufigkeit geordnet und sollen dem Benutzer das biomedizinische Wissen, wie etwa füreinander relevante Gene, verfügbar machen. Durch Anklicken der Terme können die gefundenen Dokumente angezeigt werden. Im letzten Jahr wurde nicht nur der Name des Systems vom vorherigen FACTA in FACTA+ geändert, es erfolgten auch drei bedeutende technische Neue-

¹⁷www.textpresso.org

¹⁸www.cs.man.ac.uk/~stevensr/tambis/

¹⁹refine1-nactem.mc.man.ac.uk/facta/

rungen:

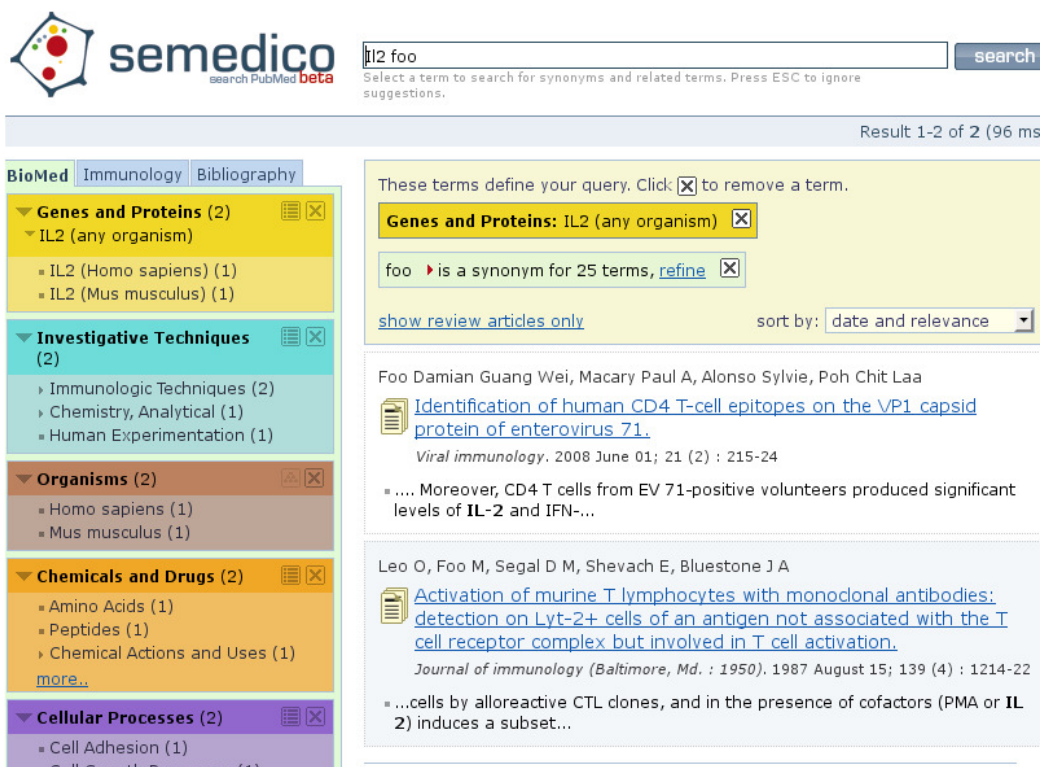
- Es wurde eine grafische Ansicht entwickelt um verwandte Terme zu verdeutlichen.²⁰ Hierfür wird eine Tree Map aus mehreren verschachtelten Rechtecke eingesetzt, deren Größe die Wichtigkeit eines Terms symbolisiert.
- Inzwischen werden auch Events, also etwa die Interaktion mehrerer Proteine miteinander, vom System erkannt. Dabei wird nur gespeichert, dass in einem Text eine bestimmte Interaktion stattfand. Die beteiligten Entitäten werden nicht beachtet und sind somit auch nicht abfragbar.
- Durch die Wahl eines Pivotterms können auch indirekt miteinander verknüpfte Ergebnisse gefunden werden. Dies entspräche etwa dem Sprung von *Hund* auf *Mensch* über *Säugetier*.

2.4 SEMEDICO

Vom JULIE LAB wurde die semantische Suchmaschine SEMEDICO im Rahmen des STEMNET Projekts entwickelt [Hahn u. a. 2007c]. In diesem sollte eine Alternative zu Suchanfragen mit Schlüsselwörtern und MeSH Termen gefunden werden, da diese sich als unzureichend und oft auch fehlerbehaftet erwiesen hatten. Mit SEMEDICO konnte eine Brücke zwischen dem Wissen um die Struktur biomedizinischer Erkenntnisse, das in Datenbanken und Ontologien abgelegt ist, und den unstrukturierten Inhalten der wissenschaftlichen Artikel geschaffen werden. Dazu werden, wie bei FACTA+, die Terme der Texte bereits im Vorfeld automatisch erkannt. Diese Terme werden dann genutzt um die Suchergebnisse zu facettieren, der Benutzer kann sie selektieren um seine Suche zu verfeinern.

Der Benutzer interagiert dabei zuerst mit einer optisch schlichten, an Google orientierten Oberfläche, in deren Suchleiste er Schlüsselwörter eingibt [Schneider u. a. 2009]. Sind diese dem System bekannt, so werden dem Benutzer interaktiv die passenden Terme vorgeschlagen. Nachdem er seine Anfrage aus Freitext und Termen zusammengestellt hat, wird er auf die in Abbildung

²⁰refine1-nactem.mc.man.ac.uk/facta-visualizer/



The image shows the SEMEDICO search interface. At the top, there is a logo for SEMEDICO (search PubMed beta) and a search bar containing the text "IL2 foo". Below the search bar, a message states: "Select a term to search for synonyms and related terms. Press ESC to ignore suggestions." The search results are displayed as "Result 1-2 of 2 (96 ms)".

On the left side, there are several vertical tabs for filtering the results: "BioMed", "Immunology", and "Bibliography". Under "Immunology", there are four main categories, each with a sub-list of terms and their counts:

- Genes and Proteins (2)**
 - IL2 (any organism)
 - IL2 (Homo sapiens) (1)
 - IL2 (Mus musculus) (1)
- Investigative Techniques (2)**
 - Immunologic Techniques (2)
 - Chemistry, Analytical (1)
 - Human Experimentation (1)
- Organisms (2)**
 - Homo sapiens (1)
 - Mus musculus (1)
- Chemicals and Drugs (2)**
 - Amino Acids (1)
 - Peptides (1)
 - Chemical Actions and Uses (1)
- Cellular Processes (2)**
 - Cell Adhesion (1)
 - Cell Growth Processes (1)

On the right side, there is a summary of the search terms: "These terms define your query. Click [X] to remove a term." It shows "Genes and Proteins: IL2 (any organism)" and "foo" (which is a synonym for 25 terms, with a "refine" link). Below this, there are two search results:

- Foo Damian Guang Wei, Macary Paul A, Alonso Sylvie, Poh Chit Laa**
[Identification of human CD4 T-cell epitopes on the VP1 capsid protein of enterovirus 71.](#)
Viral immunology. 2008 June 01; 21 (2) : 215-24
 • Moreover, CD4 T cells from EV 71-positive volunteers produced significant levels of **IL-2** and IFN-...

- Leo O, Foo M, Segal D M, Shevach E, Bluestone J A**
[Activation of murine T lymphocytes with monoclonal antibodies: detection on Lyt-2+ cells of an antigen not associated with the T cell receptor complex but involved in T cell activation.](#)
Journal of immunology (Baltimore, Md. : 1950). 1987 August 15; 139 (4) : 1214-22
 • ...cells by alloreactive CTL clones, and in the presence of cofactors (PMA or **IL 2**) induces a subset...

Abbildung 2.3: Ergebnisseite des aktuellen SEMEDICOS (Abgerufen am 22.2.2012 unter www.semedico.org/app).

4.5 gezeigte Ergebnisseite weitergeleitet. Dabei ist bisher nur eine Aneinanderreihung der Wörter in der Anfrage möglich, wie etwa *IL-2 cancer*, interpretiert als Verundung der beiden Wörter. Die Ergebnisseite bietet links die in mehrere vertikale Tabs unterteilten Facetten und mittig die Suchergebnisse. Dabei werden über den Suchergebnissen die möglichen Interpretationen der Schlüsselwörter angegeben, unter denen der Nutzer wählen kann, um dadurch seine Suche zu verfeinern.

SEMEDICOS Architektur ist bisher nur andeutungsweise publiziert [Hahn u. a. 2007c,b; Schneider u. a. 2009] und wird hier hauptsächlich auf Basis des Quellcodes beschrieben. SEMEDICO besteht grundsätzlich aus zwei Teilen, der bereits erwähnten Webseite und einer computerlinguistischen Pipeline. Bindeglied zwischen den beiden Teilen ist ein SOLR²¹ Suchserver, ein Information

²¹lucene.apache.org/solr/

Retrieval System [Smiley u. Pugh 2009]. Der Aufbau und die Funktion SEMEDICOS werden in Abbildung 2.4 veranschaulicht.

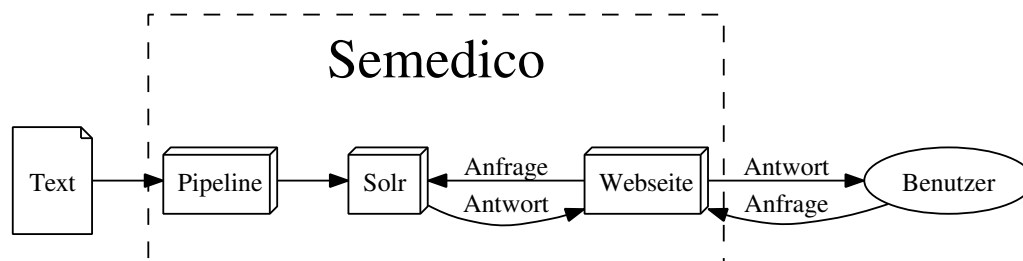


Abbildung 2.4: Aufbau und Funktion SEMEDICOS

In einer computerlinguistischen Pipeline werden dem Eingabetext in mehreren Schritten Metainformationen hinzugefügt. Abbildung 2.5 zeigt, wie aufeinanderfolgende Komponenten zuerst die Satzgrenzen (Sentence Splitter), dann die Wortgrenzen (Tokenizer) und zuletzt die Wortarten (POS-Tagger) in einem Text erkennen.

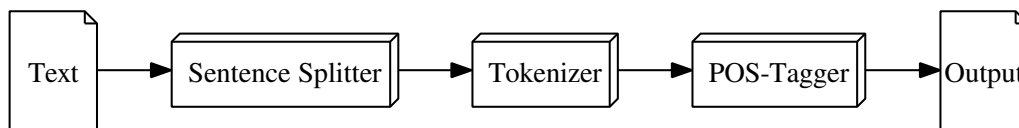


Abbildung 2.5: Beispiel für eine computerlinguistische Pipeline.

Für SEMEDICO kommen dabei auch Komponenten zum Einsatz, die syntaktische und semantische Informationen extrahieren, um etwa Proteine in einem biomedizinischen Text zu finden. JREX, ein System zur Erkennung von Relationen, das in zukünftige SEMEDICO Versionen integriert werden soll, wird unter 2.5 genauer beschrieben. SEMEDICOS Pipeline extrahiert die Informationen, indem sie MEDLINE Abstracts einliest und annotiert. Sie wurde mit dem UIMA²² (Unstructured Information Management Architecture) Framework entwickelt. UIMA ermöglicht das einfache, effiziente und modulare Zusammenfügen einzelner Komponenten zu einer Pipeline [UIMA]. Diese können die Ergebnisse anderer Komponenten nutzen, die vor ihnen angewendet wurden. So kann etwa zum Erkennen der Wortarten auf die Information

²²uima.apache.org/

zurückgegriffen werden, wo sich die einzelnen Wörter im Text befinden. Dabei stellt das Framework einen zentralen Speicher für diese schrittweise gewonnenen Metadaten zur Verfügung, die CAS (Common Analysis System).

Welche Arten von Annotationen die einzelnen Komponenten speichern oder abrufen können, wird durch das Typensystem bestimmt. Dieses beschreibt die Hierarchie der Annotationen, beispielsweise wäre ein *Organismus*²³ ein Untertyp zu einer *Biologischen Entität*, die wiederum der generellen *Named Entity* untergeordnet ist. Typen haben so genannte Features, die genutzt werden, um die jeweiligen Instanzen des Typs zu beschreiben, so kann man etwa für jede *Zeitschrift* ihren *Titel* speichern. Die Werte der Features werden nicht durch das Typensystem festgelegt, sondern während der Verarbeitung des Textes durch die einzelnen Komponenten.

Die Ergebnisse der UIMA Pipeline werden durch die LUCAS²⁴ Komponente in SOLR gespeichert [Faessler u. a. 2009]. SOLR nutzt einen Invertierten Index um die gespeicherten Informationen verfügbar zu machen. Dieser basiert auf dem gleichen Prinzip wie der Index am Ende vieler Bücher, der die für ein Schlagwort relevanten Seiten auflistet. Bei SOLR hat jedes gespeicherte Dokument²⁵ mehrere so genannte Felder, wie etwa *Autor*. Für jede Art Feld existiert ein eigener Index, dieser verweist für jeden Wert auf die relevanten Dokumente. Während man bei einem Buch mit mehreren Indices immer nur einen gleichzeitig durchsuchen kann, ermöglicht SOLR Anfragen über mehrere Felder.

Die für die Benutzer sichtbare Webseite wurde mit dem TAPESTRY²⁶ Framework erstellt. In diesem besteht jede Webseite aus TML (TAPESTRY Markup Language) und einer JAVA Klasse, einer weitgehend selbständige Organisationseinheit innerhalb eines JAVA Programms. In eine TAPESTRY Seite können so genannte Components eingebettet sein, die sich ebenfalls aus

²³Dieses und die folgenden Beispiele beschreiben die Struktur des für SEMEDICO benutzten Typensystems, aus Gründen der Lesbarkeit wurden sie übersetzt. Genaue Informationen finden sich bei Hahn et. al. [Hahn u. a. 2007b].

²⁴uima.apache.org/d/uima-addons-current/Lucas/LuceneCASConsumerUserGuide.html

²⁵Gemeint ist Dokument im Sinne des Information Retrievals, nicht der jeweilige wissenschaftliche Artikel.

²⁶tapestry.apache.org/

TML und JAVA zusammensetzen. TML ähnelt (X)HTML, es beschreibt den Aufbau einer Webseite und enthält die statischen Inhalte. JS (JAVASCRIPT) zur Erzeugung interaktiver Webseiten und CSS zur Beschreibung des Layouts können direkt in das TML eingebettet werden, typischerweise werden sie aber als separate Dateien gespeichert. JS Programme werden auf dem Rechner des Nutzers ausgeführt, sie können etwa Informationen zwischen seinem Rechner und dem Server der Webseite übertragen, ohne dass die komplette Seite neu geladen werden muss. Die JAVA Klasse dagegen sorgt für die Einbettung der auf Serverseite dynamisch generierten Inhalte, die für jede Anfrage spezifisch sind, wie etwa die Ergebnisse einer Suche. Ohne eine derartige Trennung müssten die Ergebnisse jeder möglichen Suchanfrage als fertige Webseite erstellt und gespeichert werden. Hat der Benutzer seine Anfrage eingegeben, so wird diese von der JAVA Klasse an eine Reihe weiterer JAVA Klassen weitergereicht. Diese sorgen unter anderem für eine Vorverarbeitung der Anfrage und die Kommunikation mit SOLR. Die Vorverarbeitung reduziert Wörter auf ihre Grundform und erkennt Wortgruppen wie *brain cancer*. Die vorverarbeitete Eingabe wird genutzt, um eine SOLR Anfrage zu erstellen. Bei dieser wird über mehrere Felder gesucht, um alle relevanten Dokumente zu finden. Dies ist nötig, da etwa *cancer* sowohl eine Krankheit, als auch der Name eines Autors²⁷ sein kann. Die Ergebnisse der SOLR Anfrage werden, nach einigen hier nicht relevanten Zwischenschritten, von der JAVA Klasse genutzt, um den dynamischen Teil der Webseite zu generieren, etwa die angezeigten Facetten und natürlich die Trefferliste.

2.5 JREX

JREX ist ein System zur Extraktion von Events und Relationen, etwa PPIs, aus biomedizinischen Texten [Buyko 2012; Buyko u. a. 2011, 2009; Faessler 2009]. Wie die meisten modernen computerlinguistischen Systeme nutzt JREX maschinelles Lernen [Jurafsky u. Martin 2009, Kapitel 4&7]. Dabei wird die Sprache nicht durch eine Grammatik mit fixen Regeln beschrieben,

²⁷PUBMED verzeichnet 18 entsprechende Dokumente, Stand 23.2.2012.

sondern ein statistisches Modell erstellt, dessen Parameter aus bereits annotierten Texten abgeleitet werden. Dabei muss dieses Trainingsmaterial für den Anwendungsbereich spezifisch und von fachkundigen Annotatoren erstellt sein. Durch das maschinelle Lernen kann JREX, anders als Systeme die auf regulären Ausdrücken basieren, auch Relationen erkennen, die nicht direkt im Text stehen, sondern sich nur aus dem Kontext ergeben.

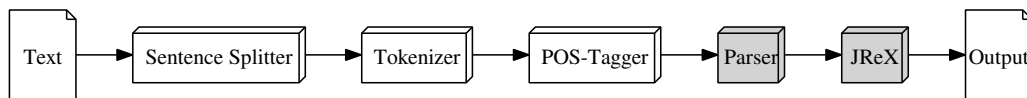


Abbildung 2.6: Computerlinguistische Pipeline mit JREX

JREX benötigt für die Relationsextraktion Informationen aus verschiedenen linguistischen Ebenen. Eine einfache Pipeline, wie sie Abbildung 2.5) zeigt, ist hierfür nicht ausreichend. Abbildung 2.6 zeigt eine erweiterte Pipeline, die neben JREX auch einen Dependenzparser enthält. Dependenzgrammatiken werden in der Computerlinguistik zunehmend eingesetzt [Jurafsky u. Martin 2009, Kapitel 12]. JREX nutzt die Dependenzparsebäume, um die Events zu erkennen, was durch die enge Verbindung zwischen Parsebaum und Satzsemantik erleichtert wird – der Satz wird auf sein Verb und dessen Argumente reduziert, alle Ergänzungen und die Binnenstruktur der Argumente fallen weg. JREX ist, wie SEMEDICO, ein mit dem UIMA Framework entwickeltes JAVA Programm. Allerdings ist es bisher nur als selbstständiges System nutzbar, soll aber in den nächsten Monaten in SEMEDICO integriert werden. Allerdings stand ein mit JREX extrahierter Datensatz zur Verfügung, der für das JenAge²⁸ (Jena Centre for Systems Biology of Ageing) Projekt erstellt worden war. Dieser wurde ersatzweise verwendet, um Speicherung und Visualisierung zu entwickeln.

²⁸www.jenage.de/

2.6 Visualisierungssoftware

Grafiken werden in biomedizinischen Lehrbüchern eingesetzt, um die Interaktionen verschiedener Substanzen zu veranschaulichen. Ein Beispiel ist die aus einem Biochemie Lehrbuch entnommenen Abbildung 2.7, die mehrere einander aktivierende Enzyme in einer Art Ablaufdiagramm zeigt.

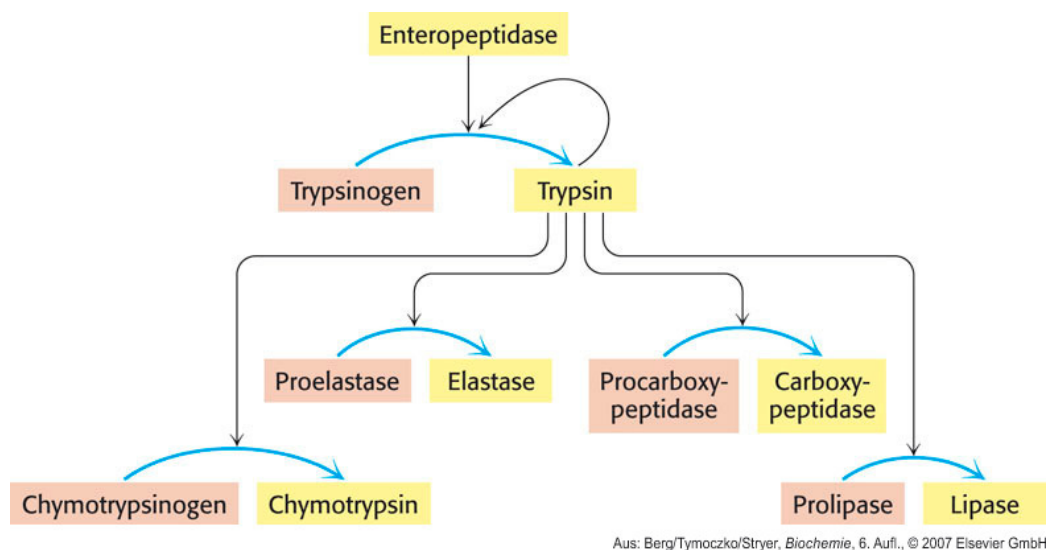


Abbildung 2.7: Biomedizinische Beispielsgrafik aus Berg u. a. [2007]

Grafiken gelten als besonders geeignet für die Informationsvermittlung, da die Augen das am höchsten entwickelte Sinnesorgan des Menschen sind und auch in großen Grafiken Details schnell und zielsicher erkannt werden können [Shneiderman 1996]. Somit sind GUIs (Graphical User Interface 'Grafische Benutzeroberflächen') besonders geeignet, um die Ergebnisse einer Suchanfrage zu präsentieren. Dabei soll eine gute GUI das schrittweise Verfeinern einer Anfrage ermöglichen, wobei Details erst bei Bedarf eingeblendet werden. Bei SEMEDICO wird dies durch die Facettierung erreicht [Schneider u. a. 2009]. Die interaktive Visualisierung komplexer Zusammenhänge gilt als besonders zur Exploration, also zum Aufspüren von Informationen, die mit den gerade behandeltem Thema nur lose verbunden sind [Fluit u. a. 2002]. Dazu werden die Informationen oft als Graph dargestellt (vgl. 2.1). Bei der Visualisierung biomedizinischer Zusammenhänge nutzt man die Knoten als

Symbole für die beteiligten Substanzen (etwa Proteine), die Kanten dagegen stehen für deren Interaktionen. Dies ist in den Abbildungen 2.8 und 2.9 erkennbar. Eine wichtige Funktion von Visualisierungssoftware ist das Filtern der Informationen in den Netzwerken, da diese im biomedizinischen Bereich äußerst umfangreich sein können. Es bietet sich eine Filterung nach Art der Interaktion und Zuverlässigkeit der Nachweismethode an. Eine derartige Filterung wird in MINT umgesetzt, Abbildung 2.9 verdeutlicht den Effekt auf den Betrachter [Ceol u. a. 2010]. Dabei zeigt der obere Teil der Abbildung die ungefilterten, der untere die gefilterten Interaktionen. Ohne die Filterung sind zu viele Details erkennbar, der Graph ist nicht zur Orientierung geeignet.

Zur Visualisierung komplexer Zusammenhänge wurden verschiedene, konkurrierende Programme entwickelt. Viele von ihnen werden heute nicht mehr unterstützt, wie etwa das für das europäische INTACT Projekt entwickelte PROVIZ, oder GUESS [Iragne u. a. 2005; Adar 2006]. Aktuell noch relevant sind die interaktiven Programme CYTOSCAPE und GEPHI [Shannon u. a. 2003; Bastian u. a. 2009]. Daneben gibt es noch Programme wie GRAPHVIZ²⁹ oder das T_EX Paket TIKZ³⁰, die zum Setzen von Diagrammen in Dokumenten dienen.

Im Moment scheint CYTOSCAPE der Standard für Visualisierung in der Biomedizin zu sein. Dies leite ich daraus ab, dass es weitaus häufiger zitiert wird als andere Programme.³¹ CYTOSCAPE ist ein in JAVA geschriebenes Open Source Programm und wurde konzipiert, um Interaktionsnetzwerke und andere biomolekulare Daten zu integrieren. Die Knoten und Kanten des Netzwerks verfügen hierzu über Attribute, die Messwerte, etwa zur Expression eines Proteins, enthalten. Zur Navigation über verschiedene Hierarchieebenen können Ontologien genutzt werden. Dabei kann der Benutzer interaktiv auf den Graph einwirken und ihn automatisch analysieren oder neu ausrichten

²⁹www.graphviz.org/

³⁰www.ctan.org/tex-archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf

³¹Der CYTOSCAPE beschreibende Artikel wird 2.388 mal zitiert, der des konkurrierenden GEPHI dagegen nur 79 mal. Für beide ermittelt mit www.scholar.google.de/ am 16.2.2012. CYTOSCAPES Vorsprung verringert sich um ca. 1000, wenn man nur den Zeitraum seit GEPHIS Veröffentlichung betrachtet.

lassen. Abbildung 2.8 zeigt einen Ausschnitt aus einem Interaktionsnetzwerk in CYTOSCAPE, unterschiedliche Farben stehen für unterschiedliche Attributswerte, wie etwa die Art einer Interaktion.

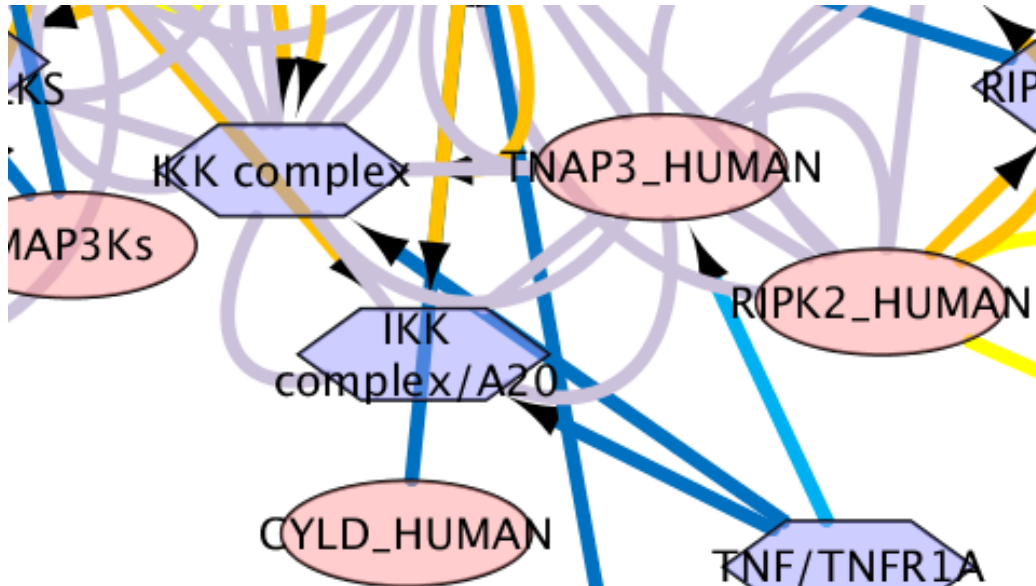


Abbildung 2.8: Interaktionsnetzwerk in CYTOSCAPE

Neben Visualisierungssoftware, die vom Nutzer heruntergeladen und installiert werden muss, gibt es auch webbasierte Lösungen, die zunehmend in biomedizinische Datenbanken integriert werden [Kerrien u. a. 2012; Tsuruoka u. a. 2011]. Dafür wird in der Regel entweder FLASH oder JS eingesetzt. Beide Programmiersprachen ermöglichen es, ein Programm vom Server zum Benutzer einer Webseite zu schicken, das dann auf dessen Rechner ausgeführt wird. Beide sind weit verbreitet, die meisten interaktiven Effekte auf Webseiten sind in JS implementiert, etwa Tooltips die erst dann sichtbar werden, wenn der Mauszeiger sich an einer bestimmten Stelle befindet. FLASH dagegen wird für die Wiedergabe von Multimediainhalten genutzt, etwa für die Filme auf Youtube³².

Für den Prototyp wurden je eine auf FLASH und eine auf JS basierte Visualisierungssoftware getestet, CYTOSCAPE WEB³³ und INFOVIS³⁴. Einige

³²www.youtube.com/

³³cytoscapeweb.cytoscape.org/

³⁴www.thejit.org

weitere webbasierte Systeme wurden von vornherein ausgeschlossen, da sie entweder proprietär waren (yFiles FLEX³⁵) oder keine ausreichenden Interaktionsmöglichkeiten boten (JSQUID³⁶). INFOVIS bietet deutlich vielfältigere Visualisierungen an, es kann anders als CYTOSCAPE WEB nicht nur Graphen darstellen, sondern etwa auch Tree Maps oder interaktive Balkendiagramme. Dies wäre von Vorteil, wenn derartige Visualisierungen später für neue SEMEDICO Funktionen gebraucht werden würden – bereits vorhandener Code und Entwicklererfahrung wären übertragbar. Dabei erwies sich INFOVIS als deutlich weniger performant, auf dem Testrechner (Ubuntu 8.04, 1GB RAM, 3.0Ghz Pentium IV, Firefox 7.01) konnten nur Graphen bis ca. 100 Kanten visualisiert werden. Bei CYTOSCAPE WEB konnten Graphen bis ca. 500 Kanten geladen werden, aber nur bis ca. 250 Kanten war eine subjektiv verzögerungsfreie Interaktion möglich.³⁷ Damit blieben beide Lösungen weit hinter dem in JAVA geschriebenen CYTOSCAPE zurück, das auf dem Testrechner auch Graphen mit mehreren tausend Kanten problemlos darstellen konnte. Somit ist CYTOSCAPE WEB zwar die bessere unter den beiden webbasierten Lösungen, kann aber, wegen der geringeren Graphengröße, Desktopsoftware nur für eine erste Übersicht ersetzen.

³⁵www.yworks.com/en/products_yfilesflex_about.html

³⁶jsquid.sbc.su.se/

³⁷Die geringe Maximalgröße der Graphen wird auch durch den veralteten Testrechner verursacht. Aktuelle Hardware sollte Netzwerke mit über 1000 Kanten flüssig darstellen können [Lopes u. a. 2010].

Kapitel 3

Technologien im Detail

Nach der Betrachtung des Forschungsstands werden in diesem Kapitel zwei für den Prototyp zentrale Technologien vorgestellt. Dabei handelt es sich um RDF (Resource Description Framework), eine Sprache zur Speicherung von Metadaten, und um die Visualisierungssoftware CYTOSCAPE WEB [Lacy 2005; Lopes u. a. 2010]. Beide sind relevant für die Entwicklung des Prototypen, da sie zur Speicherung, Abfrage und Visualisierung der PPIs eingesetzt werden.

3.1 RDF

RDF wurde ursprünglich zur Speicherung der Metadaten von Webseiten, wie etwa ihres Autors, entwickelt, kann aber zur Beschreibung aller über das Internet erreichbarer oder spezifizierbarer Dinge eingesetzt werden. RDF macht diese Informationen zwischen verschiedenen Systemen übertragbar und kann dazu an verschiedene Aufgabenstellungen angepasst werden. Die folgenden Beispiele zu Aufbau und Abfrage von RDF Datensätzen können nur einen groben Überblick bieten, weitere Details und Beispiele finden sich in den Empfehlungen des W3C¹ (World Wide Web Consortium).

¹www.w3.org/

3.1.1 Aufbau und Anfragen

RDF Datensätze sind Graphen (vgl. 2.1), in denen Knoten und Kanten über URIs (Uniform Resource Identifiers), einer Verallgemeinerung von URLs (Uniform Resource Locator), identifizierbar sind. Dabei werden die Knoten **Ressourcen**, die Kanten **Relationen** oder **Prädikate** genannt. Für die Interaktion mit dem Graph kommen Tripel aus zwei Ressourcen und einer Relation zum Einsatz. Die Tripel ähneln englischen Deklarativsätzen, wie folgendes Beispiel² zeigt:

```
Wikipedia      <http://en.wikipedia.org/wiki/Wikipedia>
was created by <http://purl.org/dc/elements/1.1/creator>
Jimmy Wales    <http://en.wikipedia.org/wiki/Jimmy_Wales>
```

Die Ressource, von der die Relation ausgeht, hier also der Wikipedia Eintrag über Wikipedia, wird **Subjekt** genannt. Ihr Gegenstück, hier der Wikipedia Eintrag zu Jimmy Wales, wird als **Objekt** bezeichnet. Das Prädikat im Beispiel verweist auf ein von der Dublin Core Metadata Initiative³ zur Verfügung gestelltes XML (Extensible Markup Language) Dokument, das das Prädikat beschreibt. XML ist eine Beschreibungssprache und besteht prinzipiell aus ineinander geschachtelten Elementen. Einzelne Elemente setzen sich aus einem öffnenden und einem schließenden Tag ('Markierung') zusammen, zwischen denen sich weitere Elemente befinden können. Elemente können Attribute haben, diese befinden sich innerhalb des öffnenden Tags. Das folgende Beispiel zeigt diese Struktur:

```
1 <beispiels_element>
2   <eingebettetes_element   beispiels_attribut="vorhanden">
3   </eingebettetes_element>
4 </beispiels_element>
```

²Die in allen Beispielen genutzten Verweise auf Wikipedia dienen nur der Illustration. In der Praxis ist Wikipedia, wegen seines sich ständig ändernden Inhalts, kein sinnvoller Bezugspunkt.

³dublincore.org/

Die Tags in den Zeilen 1 und 2 sind öffnend, die in Zeilen 3 und 4 schließend. Schließende Tags sind am Schrägstrich erkennbar. Das äußerste Element, von dem alle anderen abhängig sind, wird als Wurzelement bezeichnet. In Zeile 2 wird für das Element *eingebettetes-element* das Attribut *beispiels-attribut* mit dem Wert *vorhanden* festgelegt.

Ein RDF Objekt muss nicht zwingend eine URI sein, es kann auch ein einfacher String sein:

```
Wikipedia    <http://en.wikipedia.org/wiki/Wikipedia>
is written in <http://purl.org/dc/elements/1.1/language>
English      "en"
```

RDF ermöglicht es URIs abzukürzen, wodurch sich die Lesbarkeit der Tripel verbessert. So kann man etwa `wikipedia` als Abkürzung für `http://en.wikipedia.org/wiki/` definieren. RDF erlaubt es, das Objekt auf `rdf:nil` zu setzen und somit unspezifiziert zu lassen, dabei verweist das *rdf* Präfix auf die Sprachdefinition von RDF. Anders als in den bisherigen Beispielen werden die Tripel normalerweise in eine Zeile geschrieben, mit Abkürzungen für Dublin Core und Wikipedia ergibt sich:

```
wikipedia:Wikipedia  dc:creator  wikipedia:Jimmy_Wales
```

Ein wichtiges Werkzeug bei der Beschreibung komplexer Zusammenhänge mit RDF ist die Reifizierung [Brachman u. Levesque 2004, Kapitel 3]. Bei dieser werden keine Aussagen über einzelne URIs gemacht, sondern über ein komplettes Tripel – man trifft also eine Aussage über eine Aussage. Ein möglicher Einsatzbereich ist das Belegen der ursprünglichen Aussage. Bei der Reifizierung werden in der Regel Blank Nodes als Subjekte und Objekte genutzt. Diese werden mit einem Präfix (`_:`) erzeugt und stehen selbst nicht für eine Ressource, dienen aber als Verbindungsstück zwischen Ressourcen. Um ein Tripel zu reifizieren werden sein Subjekt, Prädikat und Objekt mit einem Blank Node verbunden, von dem aus dann Aussagen über das Tripel getroffen werden können. Diese Verbindungen werden durch die Prädikate `rdf:subject`, `rdf:predicate` und `rdf:object` hergestellt, die dem Blank Node sein Subjekt, Prädikat und Objekt zuordnen. Der Blank Node selbst

wird durch seine Verbindung über `rdf:type` mit `rdf:Statement` gekennzeichnet. Das folgende Beispiel veranschaulicht dies, indem ein Tripel reifiziert und mit einer Beschreibung versehen wird:

```
1 wikipedia:Wikipedia dc:creator wikipedia:Jimmy_Wales
2 _:blank rdf:type rdf:Statement
3 _:blank rdf:subject wikipedia:Wikipedia
4 _:blank rdf:predicate dc:creator
5 _:blank rdf:object wikipedia:Jimmy_Wales
6 _:blank dc:description "RDF-Beispiel"
```

Durch die hier vorgeführten Möglichkeiten ist RDF deutlich flexibler als XML. Es kann an beliebige Datensätze angepasst werden und erlaubt das Einbinden externer Ressourcen. RDF ist besonders geeignet um unterschiedliche Systeme zu verbinden, da RDF, über RDF SCHEMA⁴, genutzt werden kann, um die Struktur eines RDF Datensatzes festzulegen [Decker u. a. 2000]. Das Verbinden frei zugänglicher Datensätze aus unterschiedlichen Quellen wird als Linked Open Data bezeichnet und soll der nächste Schritt in der Entwicklung des World Wide Webs sein [Bizer u. a. 2009]. RDF ist die hierfür vorgesehene Sprache und soll dadurch ähnlich verbreitet und bedeutend werden wie HTML. Dabei soll RDF die Beziehungen zwischen den Datensätzen speichern und das in ihnen verwendete Vokabular verfügbar machen, die Datensätze selbst sind über ihre URIs erreichbar. Webseiten müssten so nicht die spezifischen Schnittstellen anderer Seiten implementieren, die Interoperabilität wird durch die Informationen im RDF gesichert.

Zur Abfrage und Manipulation von RDF wird die SPARQL (SPARQL Protocol And RDF Query Language) Sprache eingesetzt.⁵ SPARQL ist somit analog zum SQL Relationaler Datenbanken und ähnelt diesem auch in der Syntax. Wie bei SQL muss für eine Anfrage spezifiziert werden, was gefunden werden soll (**SELECT**), in welchem Datensatz gesucht werden soll (**FROM**) und welche Bedingung zutreffen muss, damit etwas als Antwort zurückgegeben wird (**WHERE**).

Im folgenden Beispiel sollen im Graphen *http://beispiel.graph.de* alle URIs

⁴www.w3.org/TR/rdf-schema/

⁵www.w3.org/TR/rdf-sparql-query/

gesucht werden, die als Subjekt in einem Tripel mit *bekanntes-Prädikat* und *bekanntes-Objekt* vorkommen:

```

1 SELECT  ?was
2 FROM    <http://beispiel.graph.de>
3 WHERE { ?was bekanntes-Prädikat bekanntes-Objekt }
```

Dabei können auch mehrere der mit einem Fragezeichen markierten Variablen verwendet werden, um komplexere Anforderungen zu formulieren. Werden mehrere Tripel genutzt, so müssen sie durch einen Punkt getrennt werden. Die folgende Anfrage sucht nach allen Einträgen für Sprache und Erschaffer der Wikipedia im Beispielsgraphen:

```

1 SELECT  ?wer , ?sprache
2 FROM    <http://beispiel.graph.de>
3 WHERE { wikipedia:Wikipedia  dc:creator  ?wer .
4         wikipedia:Wikipedia  dc:language ?sprache }
```

Variablen können auch mehrfach in einer Anfrage verwendet werden, die Ergebnisse müssen alle Anforderungen zugleich erfüllen. Darüber können etwa, ausgehend vom ursprünglichen Tripel, die durch Reifizierung darüber gespeicherten Metadaten gefunden werden. Das folgende Beispiel fragt die Beschreibung aus dem obigen Reifizierungsbeispiel ab:

```

1 SELECT  ?beschreibung
2 FROM    <http://beispiel.graph.de>
3 WHERE {
4         wikipedia:Wikipedia  dc:creator
5                                wikipedia:Jimmy_Wales .
6         ?blank  rdf:type  rdf:Statement .
7         ?blank  rdf:subject  wikipedia:Wikipedia .
8         ?blank  rdf:predicate  dc:creator .
9         ?blank  rdf:object  wikipedia:Jimmy_Wales .
10        ?blank  dc:description  ?beschreibung
11    }
```

SPARQL erlaubt es, über das mengentheoretische Konzept der Vereinigung, Anfragen mit Alternativen zu formulieren (**UNION**). So wählt die folgende

Anfrage alle Personen aus dem Beispielsgraphen aus, die Google oder Wikipedia gegründet haben.

```
1 SELECT  ?wer
2 FROM    <http://beispiel.graph.de>
3 WHERE {
4         {wikipedia:Wikipedia  dc:creator  ?wer }
5         UNION
6         {<http://www.google.com> dc:creator  ?wer }
7     }
```

Über eine Erweiterung von SPARQL können neue Einträge in Graphen eingefügt werden.⁶ Die Erweiterung ist noch kein Standard, die Ausführbarkeit der folgenden Anfrage kann somit nur für den unter 3.1.2 beschriebenen RDF-Server garantiert werden. Im folgenden Beispiel wird dem Beispielsgraph der Wikipediaeintrag des zweiten Wikipediagründers, Larry Sanger, hinzugefügt:

```
1 INSERT INTO <http://beispiel.graph.de>
2 {wikipedia:Wikipedia  dc:creator  wikipedia:Larry_Sanger}
```

Diese Einführung in SPARQL hat gezeigt, dass es eine mit SQL Vorkenntnissen schnell zu lernende Möglichkeit ist RDF Daten abzufragen. Alternativ könnte auch ein System mit eigener API (Application Programming Interface 'Programmierschnittstelle') zum Zugriff auf RDF Graphen genutzt werden, wie etwa SESAME⁷. Derartige Anwendungen sind allerdings weniger übertragbar, da die verwendeten Befehle nur von diesem System verstanden werden, während SPARQL als offizielle Empfehlung des W3C weite Verbreitung genießt.

3.1.2 Einsatz

Zur Speicherung von RDF-GRAPHEN gibt es verschiedenen Ansätze, die man als Triplestore bezeichnet. Obwohl RDF weitaus freiere Strukturen er-

⁶www.w3.org/Submission/SPARQL-Update/

⁷www.openrdf.org/

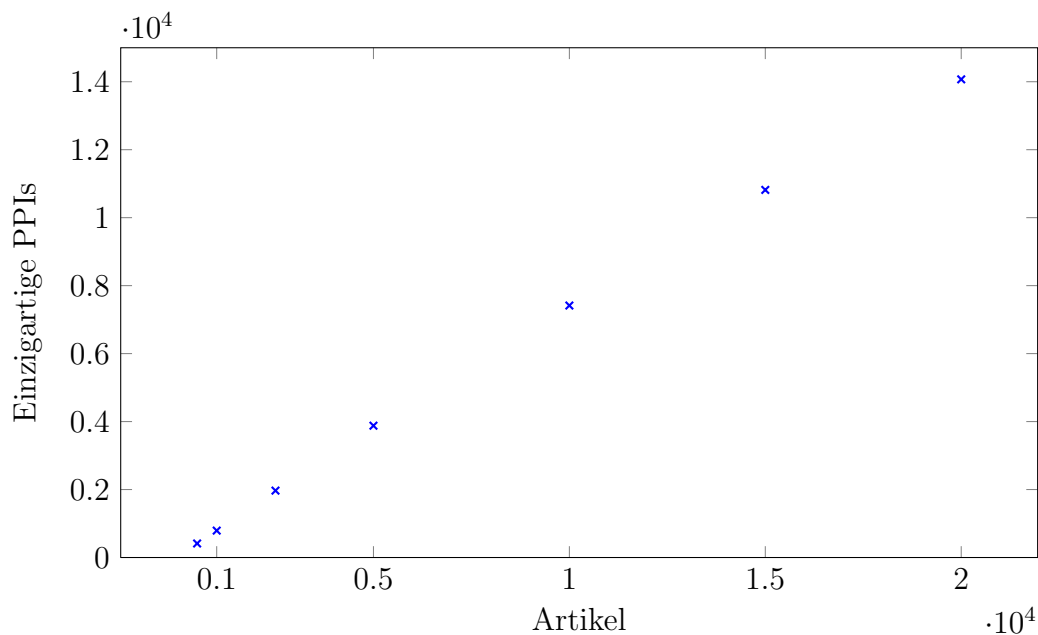


Abbildung 3.1: Verhältnis Artikel zu PPIs im Testdatensatz, Standardabweichungen unter 5%.

laubt als Relationale Datenbanken wie etwa PostgreSQL⁸, werden derartige Relationale Datenbanken oft zur Speicherung von RDF-Graphen genutzt. Die RDF spezifischen Anfragen werden über einen Wrapper ermöglicht, der zwischen SPARQL und SQL übersetzt, was allerdings mit Geschwindigkeitseinbußen einhergeht. Andere Systeme nutzen selbst entwickelte Datenbanken, die für RDF optimiert sind [Duan u. a. 2011]. Die für die Performanz der Speicherlösungen wichtige Struktur der Daten, die sich bei der Anwendung von JREX auf MEDLINE ergeben wird, kann bisher nur ausgehend von einer kleinen Testmenge aus ca. 23.000 Artikeln beurteilt werden. Die Proteine der Testmenge haben im Durchschnitt je sechs ein- und ausgehende Relationen, dies entspricht dem Verzweigungsgrad der in UNIPROT gespeicherten Daten [Duan u. a. 2011]. Basierend auf dem in Abbildung 3.1 gezeigten linearen Verhältnis der aus einem Text extrahierten einzigartigen Tripel,⁹ wären für MEDLINE lediglich 14 Millionen einzigartige PPIs zu erwarten. Da durch Reifizierung und Belege weitere Tripel hinzukommen,

⁸www.postgresql.org/

⁹Tripel mit unspezifiziertem Objekt wurden nicht gezählt.

werden insgesamt allerdings erheblich mehr Tripel zu speichern sein, bereits die Reifizierung und ein einziger Beleg mit Satz und PMID bewirken eine Verachtfachung.

Die Vielzahl der Speicherlösungen macht einen unabhängigen Vergleich nötig. Da eine eigene Messung, mangels ausreichender prozessierter MEDLINE Daten, nicht möglich war, erfolgte die Auswahl des verwendeten Triplestores anhand des BSBM¹⁰ (Berlin SPARQL Benchmark). Der BSBM wurde, wie auch andere verbreitete Benchmarks, für die zu reguläre Struktur des Benchmark Datensatzes kritisiert, die auf Relationalen Datenbanken aufbauende Lösungen bevorteilt [Duan u. a. 2011]. Diese sind den Ergebnissen aus der mit JREX prozessierten Testmenge aber in Größe und Aufbau relativ ähnlich, somit sollte eine Übertragbarkeit der Messwerte gegeben sein. Laut den von der FU Berlin veröffentlichten Ergebnissen,¹¹ ist Openlinks VIRTUOSO¹² zur Zeit der Triplestore mit der schnellsten Suche und wurde deswegen zur Implementierung des hier beschriebenen Sytsems gewählt. VIRTUOSO ist auch als Open Source Software verfügbar, die im Vergleich zur kostenpflichtigen Version fehlenden Funktionen (Redundante Server) sind für den geplanten Einsatz nicht relevant.

RDF eignet sich wegen seiner unter 3.1.1 behandelten Flexibilität gut zur Arbeit mit biomedizinischen Daten, da diese unabhängig von ihrer Quelle integriert und erweitert werden können [Cannataro u. a. 2010; Newman u. a. 2008]. Insbesondere besteht die Möglichkeit, Ontologien mit dem Datensatz zu verbinden. Hierzu wird OWL (Web Ontology Language) genutzt, eine auf RDF aufbauende Sprache [Lacy 2005]. OWL ermöglicht insbesondere das Beschreiben von mengentheoretischen Tatsachen und das Definieren von Relationen mit speziellen Eigenschaften, etwa Transitivität [Horrocks u. a. 2003]. OWL kann in mehrere Stufen unterteilt werden, die auf einen Teil der möglichen logischen Aussagen verzichten, um schnellere Verarbeitung zu ermöglichen. Durch den Einsatz von Rechnerclustern können sehr große OWL Datensätze verarbeitet werden, der Rekord übersteigt die Größe UNIPROTS

¹⁰www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/

¹¹www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/results/V6/

¹²virtuoso.openlinksw.com/

um den Faktor 50 [Urbani u. a. 2010].

Für die Entwicklung des Prototypen, wie er in dieser Masterarbeit diskutiert wird, wurde nicht auf OWL zurückgegriffen, allerdings könnte es für zukünftige Arbeiten relevant werden, etwa im Bereich Hypothesengenerierung.

3.2 CYTOSCAPE WEB

Wie unter 2.6 erläutert, ist es üblich, PPIs als Graphen zu visualisieren. Visualisierungen werden zunehmend in biomedizinische Webseiten integriert, um den Benutzern möglichst früh eine Übersicht zur Verfügung zu stellen. Unter den webbasierten Visualisierungslösungen war CYTOSCAPE WEB die Performanteste.

CYTOSCAPE WEB wurde nach dem Vorbild CYTOSCAPES entwickelt und als freie Software unter der LGPL (GNU Lesser General Public License) veröffentlicht. Seine Verbreitung ist noch gering, allerdings wird es beispielsweise seit letztem Jahr von INTACT eingesetzt [Kerrien u. a. 2012]. Im Folgenden werden einige allgemeine Informationen zu den Möglichkeiten und dem Aufbau von CYTOSCAPE WEB gegeben, Details zum Einsatz im Prototyp finden sich im Kapitel 4.3.

Obwohl der Kern von CYTOSCAPE WEB in FLASH geschrieben ist, findet die komplette Interaktion über eine JS API statt. Dazu wird in JS ein Visualisierungsobjekt erzeugt, dieses nimmt eine Repräsentation des Graphen und die sonstigen gewählten Zeichenoptionen entgegen. Zudem können Listener bei ihm angemeldet werden, wodurch etwa auf die Selektion eines Knotens reagiert werden kann.

CYTOSCAPE WEB erlaubt als Teil der Zeichenoptionen die Wahl eines Layouts und eines Stils für den Graphen. Das Layout bestimmt die Anordnung der Knoten und Kanten, der Stil ihre Darstellung. Dabei kann die Erscheinung eines Knotens oder einer Kante von dem von ihr repräsentierten Daten abhängen. Über Mapper werden dazu ihre Attribute ausgelesen und mittels vorher festgelegter Regeln auf die Visualisierung übertragen. Eine Übersicht über die vielfältigen Möglichkeiten finden sich auf der CYTOSCAPE WEB

Demo Seite¹³. Einen ersten Eindruck vermittelt Abbildung 3.2, die verschiedene Größen für Kanten und Knoten zeigt. Auch die Farben der Kanten und Knoten können frei gewählt werden, für die Abbildung wurden sie passend zum Design SEMEDICOS eingestellt. Der Graph kann CYTOSCAPE WEB in

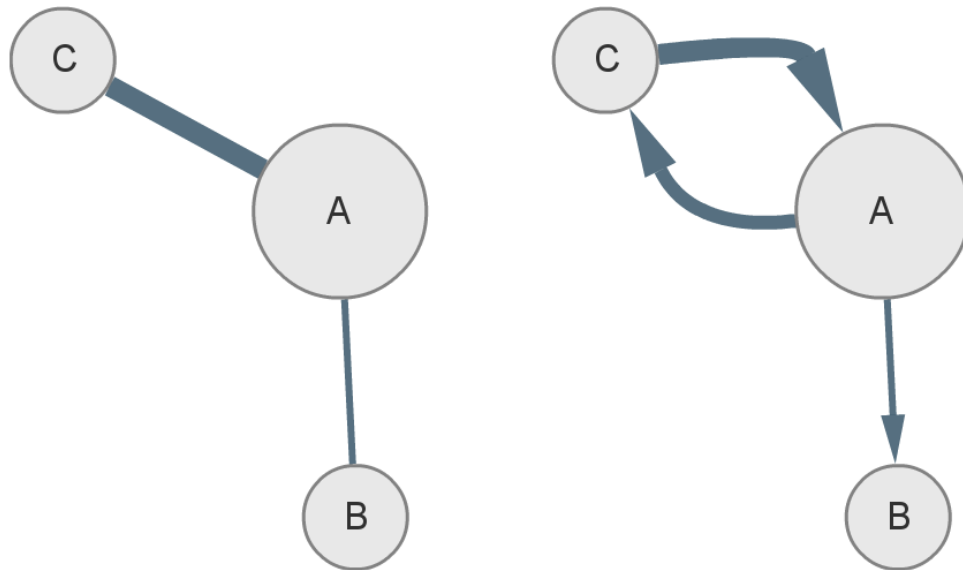


Abbildung 3.2: CYTOSCAPE WEB Darstellungsoptionen

verschiedenen Formaten übergeben werden, dabei handelt es sich um:

- NETWORKMODEL - ein JS Objekt
- GRAPHML - XML Format
- XGMML - XML Format mit Darstellungsoptionen
- SIF - minimalistisches Textformat

Dabei werden SIF¹⁴ und XGMML¹⁵ auch von CYTOSCAPE verstanden, ermöglichen es also mit einem Format die Desktopversion und die Weblösung zu unterstützen. Da SIF nur sehr einfache Graphen ohne Metainformationen erlaubt, wie sie etwa für Tooltips oder unterschiedliche Größen nötig wären, wurde für den Prototyp XGMML genutzt. Das folgende Beispiel zeigt ein

¹³cytoscapeweb.cytoscape.org/demo

¹⁴wiki.cytoscape.org/Cytoscape_User_Manual/Network_Formats/

¹⁵www.cs.rpi.edu/research/groups/pb/punin/public_html/XGMML/

minimales XGMML Dokument, das einen Graphen mit zwei Knoten und einer Kante beschreibt:

```
1 <graph label="NAME" xmlns="http://www.cs.rpi.edu/XGMML"
2     directed="1">
3   <node label="ISODC" id="node1">
4     <att type="string" name="canonicalName"
5         value="ISODC" />
6   </node>
7   <node label="SODC" id="node2">
8     <att type="string" name="canonicalName"
9         value="SODC" />
10  </node>
11  <edge label="(Binding)" source="node1" target="node2">
12    <att type="string" name="canonicalName"
13        value="ISODC (pp) SODC" />
14  </edge>
15 </graph>
```

Dabei ist `<graph>` das Wurzelement, von dem die Knoten (`<node>`) und Kanten (`<edge>`) abhängig sind. Der Verlauf der Kanten wird über ihre *source* und *target* Attribute festgelegt, die sich auf die *id* der Knoten beziehen (vgl. Zeile 11). Knoten und Kanten haben `<att>` Elemente, die Informationen zu ihnen enthalten, wie etwa *canonicalName*, das als Tooltip angezeigt wird (vgl. Zeile 3).

Kapitel 4

Implementierung

Dieses Kapitel beschreibt, wie die PPI Speicherung und Visualisierung, unter Nutzung der in den beiden vorhergehenden Kapiteln behandelten Technologien, implementiert wurde. Dafür werden, nach einigen grundsätzlichen Überlegungen, zur Integration der PPIs mit den bisherigen Funktionen SEMEDICOS, drei Probleme aufgezeigt und gelöst:

- Wie können PPIs gespeichert werden?
- Wie können die zu einer Anfrage passenden PPIs gefunden werden?
- Wie können sie visualisiert werden?

Wie unter 2.4 bereits erläutert wurde, besteht SEMEDICO bisher aus einer Pipeline, die Informationen aus biomedizinischen Texten extrahiert und einer Webseite, die diese den Nutzern zur Verfügung stellt. Beide sind über SOLR, ein Information Retrieval System, verbunden. Um PPIs speichern, suchen und darstellen zu können, sind mehrere Änderungen nötig, insbesondere muss mit dem VIRTUOSO Triplestore eine zweite Speicherlösung neben SOLR eingeführt werden. Dies ist nötig, da die PPIs sich nicht effizient in SOLRs dokumentenbasiertes Modell übersetzen lassen – dieses ist für Freitextsuche optimiert und bietet keinerlei Unterstützung für Graphenstrukturen. Zudem müssen die bisherigen Komponenten erweitert werden, damit sie auch die PPI Informationen umsetzen können und die Visualisierung muss entwickelt und mit dem restlichen System verbunden werden. Dabei müssen die alten und neuen Komponenten miteinander interagieren können, aus der Visuali-

sierung soll die Suche nach Artikeln möglich sein und umgekehrt. Abbildung 4.1 zeigt das zur PPI Verarbeitung erweiterte SEMEDICO.

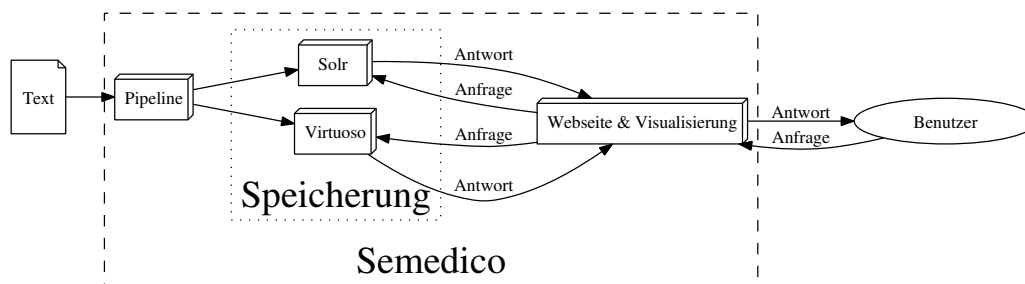


Abbildung 4.1: Aufbau des erweiterten SEMEDICOS

4.1 Speicherung

Die aus den Texten extrahierten PPIs müssen, um sie für IR nutzen zu können, aus der UIMA CAS (vgl. 2.4) in geeignete Systeme übertragen werden. Dabei wird SOLR genutzt, um für jeden Text zu speichern, welche PPIs in ihm vorkommen. VIRTUOSO dagegen enthält die Proteine und ihre Interaktionen als RDF-Graph, dazu kommen die PMIDs der Texte und der Satz in dem die PPI gefunden wurde als minimale Metadaten (vgl. 2.2 und 3.1). An dem hierfür benutzten Blank Node könnte man auch weitere Metadaten speichern, die zur Filterung der Ergebnisse geeignet wären, etwa Angaben zur Konfidenz der Extraktion oder den Impact Factor der Veröffentlichung. Zur Übertragung der Daten aus der CAS sind spezielle UIMA Komponenten nötig. Diese können über UIMAs API aus der CAS alle Annotationen eines Typs erhalten und müssen diese dann in ein mit SOLR und VIRTUOSO kompatibles Format übertragen. Die Entwicklung dieser Komponenten erwies sich aus zwei Gründen als problematisch:

1. JREX wurde bisher noch nicht in die SEMEDICO Pipeline integriert.
2. Der JREX Output aus dem JenAge Projekt nutzt neben *agent-patient* Annotationen auch *theme-cause* Annotationen. Diese uneinheitliche

Struktur ist nicht direkt in RDF übertragbar und würde das Erstellen von Suchanfragen erschweren.

Um diese beiden Probleme zu lösen, wurden mehrere UIMA Komponenten eingesetzt:

1. Um die Erweiterung SEMEDICOS zu simulieren, liest der in UIMA enthaltene XMI READER den bereits vorhandenen JREX Output (vgl. 2.5) ein und speichert ihn in der CAS. Für nach ihm laufende Komponenten ist dies nicht von einer Pipeline zu unterscheiden, die diese Informationen direkt extrahiert hat.
2. Die einheitliche Reigenfolge der Proteine und PPIs wird über eine neue Komponente erreicht. Dieser ARGUMENTREORDERER vertauscht *theme* und *cause*. Um die Kompatibilität mit anderen Komponenten nicht zu gefährden, wurde der Typ RELATIONMENTION, der für eine im Text gefundene Relation mit ihren Argumenten steht, dafür um das Feature REORDEREDARGUMENTS erweitert, die ursprüngliche Anordnung im Feature ARGUMENTS bleibt erhalten.

Zwei UIMA Komponenten, der TRIPLEINTOTYPESWRITER und LUCAS wurden eingesetzt, um die nun korrekt geordneten Daten in ein für SOLR verständliches Format zu konvertieren. LUCAS erzeugt direkt einen LUCENE¹ Index, das intern von SOLR verwendete Speicherformat. Da LUCAS dabei allerdings nur auf einen Teil der im Typensystem gespeicherten Informationen zugreifen kann, das Erkennen von Unterklassen ist nicht möglich, wurde der Typ ARGUMENTMENTION, der für ein Argument einer PPI steht, um das Feature RELATIONSTRING erweitert. In diesem werden die interagierenden Proteine und die Art ihrer Interaktion in dem an RDF angelehnten Format (*protein1-interaktion-protein2*) gespeichert. Für die Zuordnung der UIMA Typen auf die SOLR Felder (vgl. 2.4) benötigt LUCAS ein XML Dokument, das so genannte MAPPING FILE, als Anleitung. Das folgende Beispiel zeigt einen Ausschnitt des zum vorhandenem JREX Output passenden MAPPING FILES:

¹lucene.apache.org/

```

1 <fields>
2 [...]
3 <field name="PPI" index="yes"
4     stored="yes" termVector="no">
5     <annotations>
6         <annotation sofa="_InitialView"
7             type="de.julielab.jules.types.ArgumentMention">
8             <features>
9                 <feature name="relationString" />
10            </features>
11        </annotation>
12    </annotations>
13 </field>
14 [...]
15 </fields>

```

Der Inhalt des beschriebenen Feldes, PPI, wird gespeichert (**stored="yes"**) und indiziert (**index="yes"**). Somit können die PPIs zur Suche nach einem Dokument² genutzt werden, können aber auch für jedes, wie auch immer gefundene, Dokument angezeigt werden. In das Feld wird für jede ARGUMENTMENTION der Wert ihres RELATIONSTRINGS gespeichert. Da die ARGUMENTMENTION Annotation sich auf die im Text vorkommenden Proteine bezieht und LUCAS auch deren Position in den Index schreibt, können dadurch die PPIs im Text hervorgehoben werden.³ Neben den Änderungen an der UIMA Pipeline musste auch das SOLR Schema überarbeitet werden. In dieser Konfigurationsdatei sind alle SOLR bekannten Felder aufgelistet, hier musste das neue Feld PPI hinzugefügt werden.

Durch diese Änderungen ist es möglich, Dokumente über ihre PPIs in SOLR zu suchen, wobei dessen Unterstützung für Wildcards auch unscharfe Suchen erlaubt. Zudem können für beliebige Dokumente, etwa die über eine klassische SEMEDICO Suche gefundenen Artikel, die in ihnen enthaltenen PPIs abgefragt werden.

²Wieder im Sinn des IR.

³Dies ist eine Funktion SOLRs, die auch in SEMEDICO genutzt wird. Der Prototyp unterstützt es, wegen der ihm zu Grunde liegenden SEMEDICO Version, nicht.

Während die Änderungen an SOLR nur zur Integration der PPIs in die bisherigen Fähigkeiten SEMEDICOS dienen, ermöglicht VIRTUOSO die neu hinzugekommene Visualisierung. Die Kommunikation der Pipeline oder Webseite mit VIRTUOSO erfolgt über JDBC⁴, eine standardisierte Schnittstelle für die Interaktion zwischen JAVA Programmen und Relationalen Datenbanken. Für VIRTUOSO existieren JDBC Treiber, die auch zur Übertragung von SPARQL genutzt werden können.⁵ Dabei wird die Anfrage in SPARQL formuliert, die Antwort besteht aus einer Tabelle mit den gesuchten Variablen als Spalten.

Um die PPIs aus der UIMA CAS in VIRTUOSO zu übertragen, wird eine UIMA Komponente, der EVENTODBWRITER, genutzt. Diese erfragt zuerst alle RELATIONMENTIONS aus der CAS und extrahiert aus diesen Subjekt, Objekt und Prädikat für ein RDF Tripel. Zudem werden von der CAS die PMID des Dokuments und der Satz in dem die PPI gefunden wurde erfragt. Diese Informationen werden später, durch die unter 3.1.1 erklärte Reifizierung, als Tripel mit dem PPI Tripel verbunden. Die Tripel werden nun stapelweise genutzt, um einen SPARQL String zu vervollständigen, der über JDBC an VIRTUOSO übertragen werden kann, womit der Speichervorgang abgeschlossen ist.

4.2 Zugriff auf die Relationen

Die in SOLR und VIRTUOSO gespeicherten Relationen werden beide auf unterschiedliche Arten abgefragt. Dabei sind die SOLR Anfragen einfacher strukturiert als die VIRTUOSO Anfragen. Allerdings musste, um das Abfragen von PPIs über SEMEDICOS Suche zu ermöglichen, die Vorverarbeitung der Benutzereingaben drastisch geändert werden.

⁴www.oracle.com/technetwork/java/overview-141217.html

⁵docs.openlinksw.com/virtuoso/VirtuosoDriverJDBC.html

4.2.1 Parser und SOLR-Query

Wie unter 2.4 bereits erwähnt, kann in SOLR der Inhalt mehrerer Felder zugleich als Suchkriterium genutzt werden. Dies wird von SEMEDICOS QUERY-TRANSLATIONSERVICE genutzt, um die Benutzereingabe auf die passenden Felder zu verteilen. Dieser ist, wie alle Services, permanent aktiv und stellt den Webseiten allgemeine Funktionen zur Verfügung.

SOLRs Syntax für die Suche in Feldern ist simpel, es wird ein Doppelpunkt zwischen Feld und Wert gesetzt. Bisher wurden in den Benutzereingaben alle Wörter als gleichwertig angesehen und verundet. Davor werden vom QUERY-DISAMBIGUATIONSERVICE Terme in der Eingabe erkannt, diese werden nur in einigen wenigen Feldern gesucht, Freitext dagegen in allen. Das folgende Beispiel einer SOLR Anfrage zeigt, wie die Benutzereingabe *il-2* als Term *IL2* erkannt und dieser in verschiedenen Feldern, wie etwa *title*, gesucht wird:

```
( title:IL2 OR text:IL2 OR mesh:IL2 )
```

Die PPI Informationen erfordern eine Erweiterung dieses Formats, es muss auch im Feld *PPI* gesucht werden. Dazu werden erkannte Interaktionen, wie etwa die Bindung zweier Proteine, mit der bisherigen SOLR Anfrage verodert. Dies zeigt die folgende Anfrage für die Benutzereingabe *x Binding y*, in der *x* nicht nur in den bisherigen Feldern, wie etwa *title*, gesucht wird, sondern auch in die Bedingung für das Feld *PPI* integriert ist:

```
( PPI:x-Binding-y ) OR (( title:x OR text:x [...] ) )
```

Zur Erkennung der Interaktionen in einer Benutzeranfrage werden ein Parser und ein Tokenizer eingesetzt, beide sind nur zur Verarbeitung der Benutzereingaben geeignet. Durch sie können auch komplexe Benutzereingaben, die gleichzeitig nach einer PPI und anderen Dingen suchen, verarbeitet werden. Zudem erlaubt diese Art der Verarbeitung eine freiere Formulierung der Anfragen; der Benutzer ist etwa nicht gezwungen, Proteine und Interaktionen mit Bindestrichen zu verbinden, auch wenn SOLR dies für die PPIs erwartet. Da der Parser die durch Klammern ausgedrückte Hierarchie der Benutzereingabe versteht, könnte er auch genutzt werden, um den QUERY-DISAMBIGUATIONSERVICE zu deren Verarbeitung zu befähigen.

Der Tokenizer wurde aus einer Vorlage mit JFLEX⁶ erzeugt. Er erkennt die Tokens anhand regulärer Ausdrücke und stellt dem Parser Symbole zur Verfügung. Symbole entsprechen Wortarten, sind allerdings sehr eingeschränkt und auf die formale Sprache spezialisiert. Diese umfassen etwa *ALPHANUM* (generische Zeichenfolgen), *OR* (logisches oder) und natürlich *RELATION* (beliebige Relation). Durch die regulären Ausdrücke werden dabei sehr unterschiedliche Zeichenketten auf ein Symbol reduziert. *AND* wird etwa für folgende Eingaben emittiert:

"And" | "and" | "AND" | "&" | "&&"

Da der automatisch generierte Tokenizer strikt an Leerzeichen trennt, ist eine Nachbearbeitung der Symbole nötig. Dies geschieht durch den COMBININGLEXER, der benachbarte Symbole, die gemeinsam einen Term ergeben, zusammenlegt. Ist etwa *heart_attack* als Term vorhanden, so werden die Symbole *heart* und *attack* zum Symbol *heart_attack* kombiniert.

Um die Struktur der Anfrage zu verstehen, wird aus den Symbolen ein Parsebaum erstellt. Der dazu verwendete Parser wurde von Hand geschrieben, da hierdurch eine besonders robuste Implementierung möglich ist, die auch unvollständige Eingaben akzeptiert. Dies könnte genutzt werden, um den Benutzer darauf hinzuweisen, dass er etwa eine Klammer mehr geöffnet als geschlossen hat und ihm gleichzeitig Ergebnisse für die vermutliche Anfrage zu zeigen – aktuell wird lediglich das Fehlen von *ANDs* zwischen den einzelnen Wörtern der Eingabe ausgeglichen. Der Parsebaum setzt sich aus verschiedenen Knoten zusammen, die alle von der Oberklasse *NODE* abgeleitet sind. Die Klasse *PARSETREE* fungiert als Wrapper um den Parsebaum und regelt die Interaktion der Knoten mit anderen Objekten. Die Klassen der Knoten sind dabei spezialisiert, um die Regeln der Grammatik durchzusetzen. So kann etwa ein *NOTNODE* (steht für die logische Negation) nur ein einziges Kind haben, Knoten können hinzukommende Knoten an ihren Nachfahren weiterreichen und Knoten versuchen hinzukommende Knoten möglichst tief und möglichst links im Parsebaum einzufügen. Dies vereinfacht das Hinzufügen neuer Knotenklassen mit eigenen Regeln, wie etwa eines Knoten mit drei

⁶www.jflex.de/

Kindern. Zum Parsen wird ein Top-Down Parser eingesetzt:

```

1 recursiveParse(){
2   root = null
3   token = nextToken()
4   while (token != null && token != ")") {
5       if (root == null){
6           if (token == "(")
7               root = recursiveParse()
8           else
9               root = token
10      }
11      else if (root.subtreeCanTakeNode()){
12          //Knoten in Teilbaum einfügen
13      }
14      else{
15          if (token == OR)
16              root = new BinaryNode(OR, root, null);
17          else if (token == AND)
18              root = new BinaryNode(AND, root, null);
19          // ab hier implizites und
20          else if (token == "(")
21              root = new BinaryNode(AND, root,
22                                     recursiveParse());
23          else
24              root = new BinaryNode(AND, root, token);
25      }
26      token = nextToken()
27  }
28  return root;
29 }
```

Dieser Parser erstellt etwa für die Benutzereingabe x *OR NOT* ($y z$) den Parsebaum in Abbildung 4.2, indem nacheinander die Anweisungen der Zeilen 9, 16, 9 und 24 ausgeführt werden. Dabei erfolgt zwischen den Zeilen 16 und 9, über den in Zeile 12 angedeuteten Code, ein Rekursionsschritt.

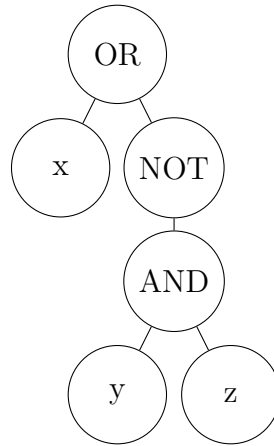


Abbildung 4.2: Beispiel für einen Parsebaum

Der Parsebaum kann nun genutzt werden, um korrekte SOLR Anfragen zu formulieren, wie im Beispiel *x Binding y* aus Abschnitt 4.2.1 – dazu muss lediglich der Parsebaum wieder in eine flache Form überführt werden, es ergibt sich *x OR NOT (y AND z)*.

4.2.2 SPARQL in SEMEDICO

Während in den vorherigen Abschnitten der Zugriff auf Artikel mit bestimmten PPIs behandelt wurde, wird nun die Suche nach Interaktionen zwischen den Proteinen beschrieben. Dazu wird die SPARQL Abfragesprache, die unter 4.1 bereits vorgestellt wurde, genutzt, um die in VIRTUOSO gespeicherten Tripel zu durchsuchen. Dies geschieht über den `RDFSEARCHSERVICE`, der die jeweils gesuchten Variablen in ansonsten fertige SPARQL Anfragen einsetzt. Um alle Interaktionen zu erhalten, an denen ein Protein beteiligt ist, muss man nach den Tripeln suchen, in denen es entweder als Subjekt oder Objekt vorkommt. Diese Tripel können dann vereinigt werden, um über ihre Reifizierungen auch die jeweiligen Belege zu finden. Dies zeigt das folgende Beispiel, in dem *gesuchtesProtein* und *durchsuchterGraph* durch ihre jeweiligen Werte zu ersetzen sind:

```

1 SPARQL SELECT ?s ?p ?o ?pmid ?sentence

```

```

2 FROM durchsuchterGraph WHERE {
3   {
4     uniprot:gesuchtesProtein ?p ?o.
5     ?x rdf:subject uniprot:gesuchtesProtein
6     ?x rdf:object ?o.
7     ?p rdfs:subClassOf <http://placeholder/ppi/>
8   } UNION {
9     ?s ?p uniprot:gesuchtesProtein.
10    ?x rdf:object uniprot:gesuchtesProtein.
11    ?x rdf:subject ?s.
12    ?p rdfs:subClassOf <http://placeholder/ppi/>
13  }
14  ?x rdf:predicate ?p.
15  ?x rdf:type rdf:Statement.
16  ?x <http://placeholder/has-pmid> ?pmid.
17  ?x <http://placeholder/has-text/> ?sentence.
18 }

```

Dabei sollte *durchsuchterGraph* konstant bleiben, *gesuchtesProtein* dagegen hängt von der jeweiligen Anfrage ab. Zeilen 7 und 12 zeigen den Einsatz von RDF SCHEMA um nur nach Tripeln zu suchen, deren Prädikat für eine PPI steht. Dazu wurden dem Graphen, für Interaktionen wie *Binding*, zuvor die entsprechenden Tripel hinzugefügt. Ohne diese Einschränkung würde auch die Verbindung des Proteins mit seinem (zur Reifizierung genutzten) Blank Node in der restlichen Suche genutzt. URIs wie `http://placeholder/has-pmid` in Zeile 16 sind nur für den internen Gebrauch geeignet. Sollen die RDF Daten anderen Forschern direkt zugänglich gemacht werden, so müssen sie durch gültige URIs, die zu einer XML Beschreibung des Prädikats führen, ersetzt werden. Sollte später Bedarf an einer serverseitigen Filterung der Ergebnisse bestehen, da diese zu umfangreich werden um sie visualisieren zu können, so müsste man bei der SPARQL Anfrage alle Tripel ausschließen, deren (per Reifizierung gespeicherte) Konfidenz unter einem Schwellenwert liegt.

Es existieren zwei Varianten der obigen Anfrage: Die Erste schließt Tripel aus, deren Objekt `rdf:nil` ist, da einwertige PPIs durch ihre hohe Anzahl

die Übersichtlichkeit der Visualisierung reduzieren. Die Zweite nutzt zudem das Schlüsselwort **distinct**, um nur ein Tripel für jede PPI zu finden. Sie verzichtet auch auf das Abfragen der Reifizierungsknoten und Belege und ermöglicht es den Benutzern sich einen ersten Überblick über einen großen Datensatz zu verschaffen.

Um größere Ausschnitte eines PPI-Netzwerks zu untersuchen, müssen nicht nur die PPIs gefunden werden, an denen ein Protein direkt teilnimmt, sondern auch alle, an denen seine Interaktionspartner teilnehmen. So würde die obige Anfrage für das folgende Beispiel bei der Suche nach *a* lediglich die PPIs mit *b* und *c* finden:

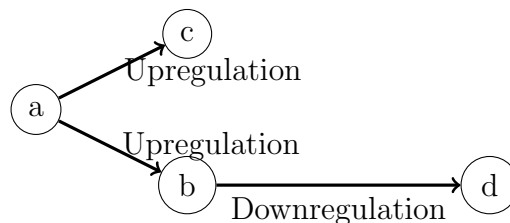


Abbildung 4.3: Beispiel für einen PPI Graphen

Will man auch indirekt mit *a* verbundene Proteine, wie etwa *d* finden, so lässt sich dies durch mehrmaliges Ausführen der Anfrage erreichen. Dazu wird jedes Protein, das als Partner von *a* gefunden wurde, als *gesuchtesProtein* in die SPARQL Vorlage eingesetzt und die Ergebnisse aller Anfragen werden kombiniert. Eine direkte Möglichkeit, alle Knoten bis zu einer gegebenen Pfadlänge abzufragen, gibt es in SPARQL leider nicht. VIRTUOSO implementiert die eigentlich hierfür gedachten Path Expressions, erlaubt dabei aber keine unspezifizierten Prädikate. Die über JDBC übermittelten Ergebnisse der Anfrage werden vom RDFSEARCHSERVICE in XGMML umgewandelt, das von CYTOSCAPE und CYTOSCAPE WEB gelesen werden kann, um die Interaktionen zu visualisieren.

4.3 Anzeigen der Ergebnisse

Die durch SOLR und SPARQL erhaltenen PPI Informationen werden den Benutzern auf zwei unterschiedliche Arten angeboten. Einerseits werden sie in die bisherigen Ergebnisseiten SEMEDICOS (vgl. 2.4) eingebunden, andererseits werden sie auf einer neu hinzugekommenen Seite mit CYTOSCAPE WEB visualisiert. Zwischen der neuen Visualisierungsseite und der bisherigen Ergebnisansicht kann dabei über Links gewechselt werden.

Um dies zu erreichen, wurde in die Ergebnisansicht ein Link eingefügt, der die Visualisierung aller PPIs in diesen Artikeln startet. Dazu die Komponente QUERY-PANEL der Seite MAIN um einen PAGELINK erweitert. PAGELINKS öffnen neue Seiten und können diesen, durch Anhänge an die URL, Kontextinformationen übergeben. Der PAGELINK startet die Visualisierung und teilt ihr die von SOLR gefundenen PPIs mit. Abbildung 4.4 zeigt die modifizierte Webseite, der Link ist in Rot markiert.

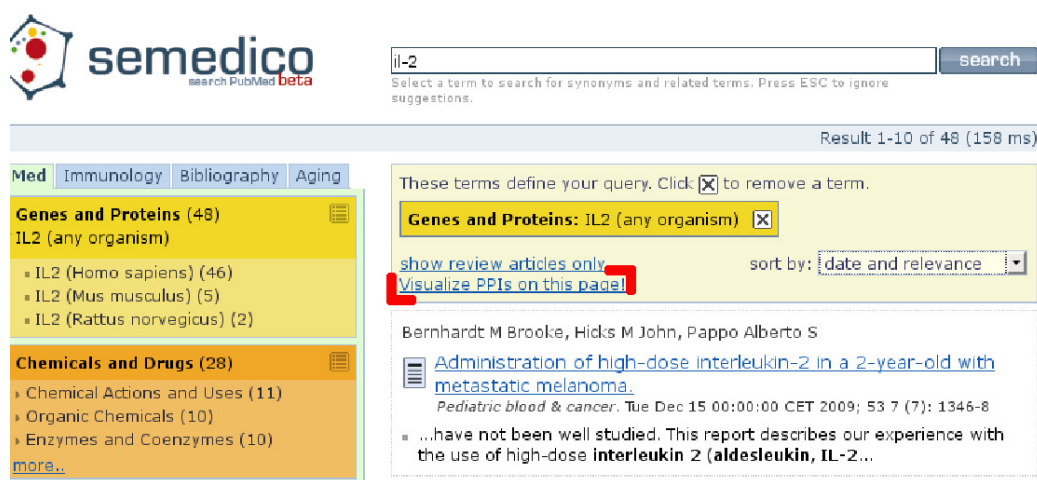


Abbildung 4.4: SEMEDICO Ergebnisseite mit hervorgehobenem PPI Link.

Zum Vergleich zeigt Abbildung 4.5 nochmals die bisherige Ergebnisansicht. Analog wurden, durch Änderungen an der ARTICLE Komponente, in der Ansicht der einzelnen Artikel Links zu den von SOLR für diesen Artikel gefundenen PPIs eingefügt.

Die Visualisierung wurde als NETWORKVIEWER, eine neu geschriebene TA-

semedico
search PubMed **beta**

IL2 foo search

Select a term to search for synonyms and related terms. Press ESC to ignore suggestions.

Result 1-2 of 2 (96 ms)

BioMed Immunology Bibliography

Genes and Proteins (2)

- IL2 (any organism)
 - IL2 (Homo sapiens) (1)
 - IL2 (Mus musculus) (1)

Investigative Techniques (2)

- Immunologic Techniques (2)
- Chemistry, Analytical (1)
- Human Experimentation (1)

Organisms (2)

- Homo sapiens (1)
- Mus musculus (1)

Chemicals and Drugs (2)

- Amino Acids (1)
- Peptides (1)
- Chemical Actions and Uses (1)
- [more..](#)

Cellular Processes (2)

- Cell Adhesion (1)
- Cell Growth Processes (1)

These terms define your query. Click ☐ to remove a term.

Genes and Proteins: IL2 (any organism) ☐

foo ☒ is a synonym for 25 terms, [refine](#) ☐

[show review articles only](#) sort by: date and relevance

Foo Damian Guang Wei, Macary Paul A, Alonso Sylvie, Poh Chit Laa

[Identification of human CD4 T-cell epitopes on the VP1 capsid protein of enterovirus 71.](#)

Viral immunology. 2008 June 01; 21 (2) : 215-24

• Moreover, CD4 T cells from EV 71-positive volunteers produced significant levels of **IL-2** and IFN-...

Leo O, Foo M, Segal D M, Shevach E, Bluestone J A

[Activation of murine T lymphocytes with monoclonal antibodies: detection on Lyt-2+ cells of an antigen not associated with the T cell receptor complex but involved in T cell activation.](#)

Journal of immunology (Baltimore, Md. : 1950). 1987 August 15; 139 (4) : 1214-22

• ...cells by alloreactive CTL clones, and in the presence of cofactors (PMA or **IL 2**) induces a subset...

Abbildung 4.5: Ergebnisse des aktuellen SEMEDICOS (Abgerufen am 22.2.2012 unter www.semedico.org/app).

PESTRY Seite, implementiert. Sie besteht aus dem interaktiven PPI Graph und mehreren Schaltelementen für Darstellungsoptionen. Wenn Knoten oder Kanten selektiert werden, so erscheinen unter dem Graph weitere Informationen zu ihnen.

Bei jeder TAPESTRY Seite führt der JAVA Teil einer Seite beim Laden die ONACTIVATE Methode aus. Beim NETWORKVIEWER liest diese den per PAGELINK übergebenen Kontext aus und nutzt ihn, um über den RDFSEARCH-SERVICE (vgl. 4.2.2) das PPI Netzwerk als XGMML abzurufen. Dabei wird, ausgehend von der Anzahl der abgefragten PPIs, entweder die detaillierte Ansicht oder die Übersicht abgefragt. Das XGMML wird zusammen mit dem Rest der Seite an den Rechner des einzelnen Benutzers gesandt. Dort wird, durch in der Seite eingebettetes JS, das XGMML in CYTOSCAPE WEB geladen – der Benutzer kann nach einer kurzen Wartezeit mit dem Graph

interagieren. Abbildung 4.6 zeigt einen Graph für PPIs im Übersichtsmodus, zwischen allen Proteinen befindet sich maximal je eine Kante, die keinerlei Details zur Art der Relation enthält.

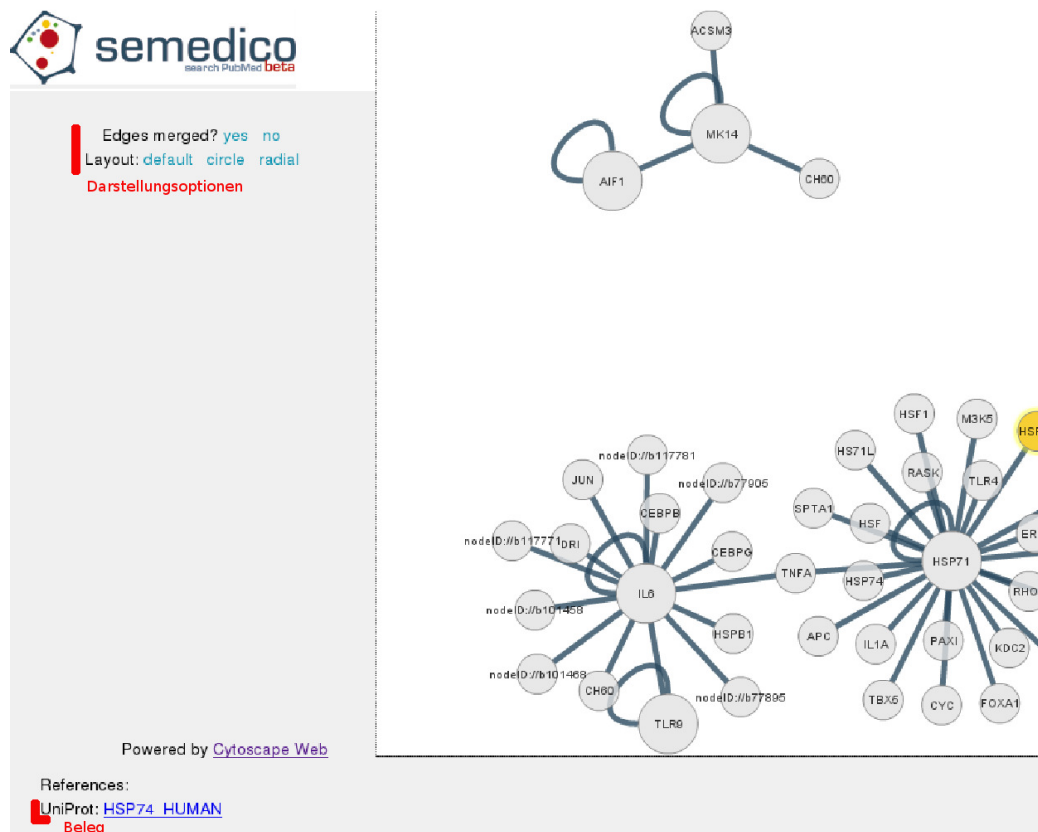


Abbildung 4.6: PPI Visualisierung im Übersichtsmodus mit Hervorhebungen. Da die PPI Extraktion für MEDLINE noch nicht durchgeführt wurde, basiert die Abbildungen auf Platzhaltern.

Unten links in der Abbildung, in Rot markiert, ist dynamisch erzeugter Text erkennbar. Dieser wurde durch einen Listener als Reaktion auf die Auswahl des orange hervorgehobenen Proteins erzeugt. Der im Text enthaltene Link führt zur UNIPROT Seite des ausgewählten Proteins. Oben links befinden sich Links, die über Listener die Darstellungsoptionen des Graphen ändern können, dadurch können etwa die Kanten zusammengelegt werden, wie in Abbildung 3.2 gezeigt.

Durch Doppelklick, oder über das per Rechtsklick gestartete Kontextmenü,

kann der Benutzer ein Protein auswählen und dessen Interaktionsnetzwerk über AJAX (Asynchronous JavaScript and XML) anfordern. Dazu wird vom dafür zuständigen JS Listener auf dem Rechner des Benutzers eine Anfrage an den die Webseite zur Verfügung stellenden Server gesandt, wo die LOAD-NETWORK Methode des NETWORKVIEWERS ausgeführt wird. Diese nutzt den RDFSEARCHSERVICE, um die Interaktionen um das gewählte Protein zu ermitteln und sendet sie als XGMML an den Rechner des Benutzers. Dort wird das XGMML vom Listener entgegengenommen und an CYTOSCAPE WEB übergeben, der in FLASH animierte Graph wird neu geladen und steht dem Benutzer zur Verfügung. Abbildung 4.7 zeigt die detaillierte PPI Visualisierung für ein Protein.

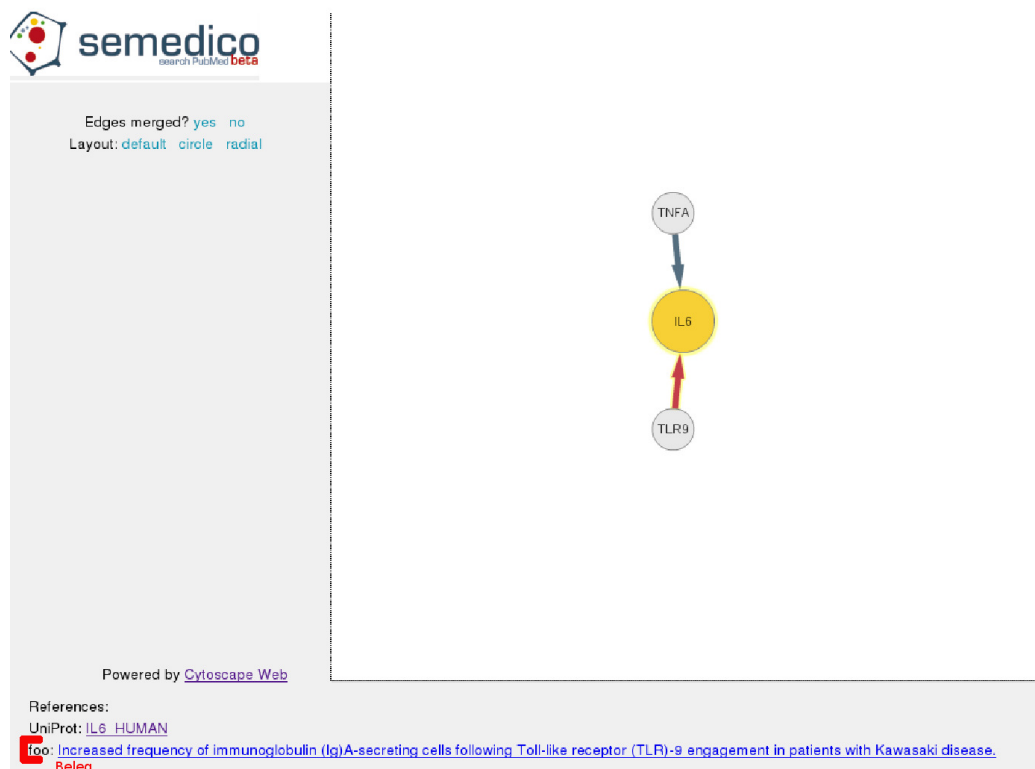


Abbildung 4.7: PPI Visualisierung mit Details und Hervorhebungen.

Für diese Abbildung wurde im Graph nicht nur ein Protein selektiert, sondern auch eine Relation. Für diese wird der genaue Name, im Beispiel der Platzhalter *foo*, im rot markierten Bereich angezeigt. Hinter dem Namen der

Relation befindet sich ein Link, der diesmal nicht zu UNIPROT führt, sondern zur SEMEDICO Ansicht des Artikels, in dem die PPI gefunden wurde. Gibt es für eine PPI mehrere Belege, so werden diese untereinander aufgelistet. Abbildung 4.8 fasst den Ablauf der Interaktion mit dem SEMEDICO Prototyp nochmals zusammen. Die Pfeile stehen für die Verlinkungen zwischen den Seiten und die Möglichkeit per AJAX neue Inhalte in die Visualisierung zu laden.

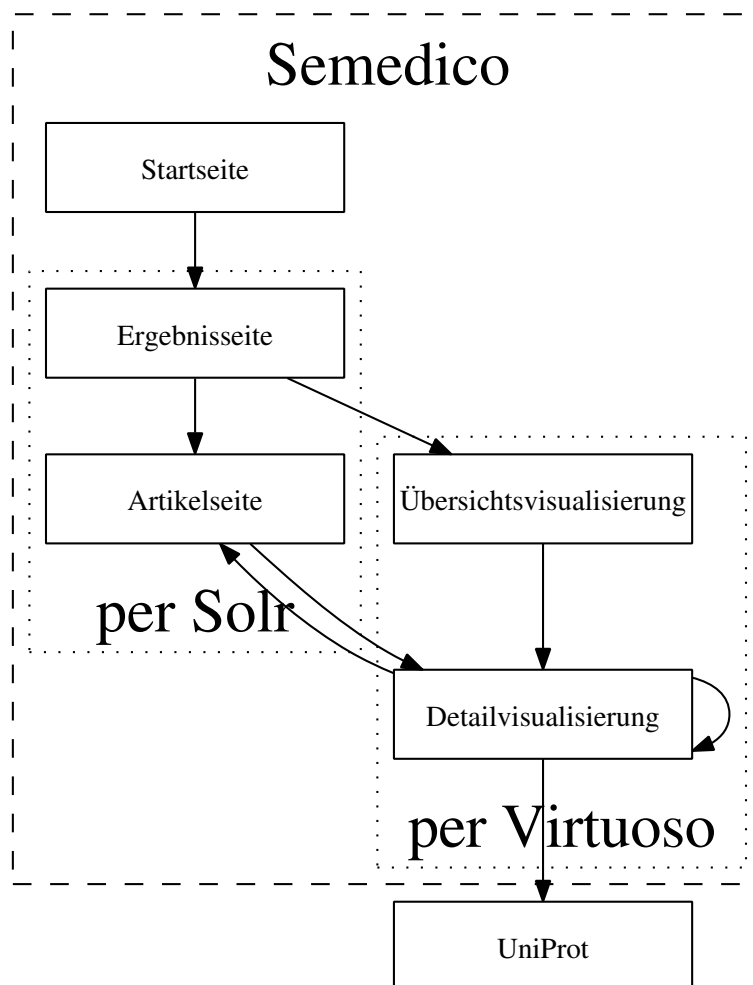


Abbildung 4.8: Interaktion mit dem Prototyp

Kapitel 5

Zusammenfassung und Ausblick

Diese Arbeit hat gezeigt, wie SEMEDICO für die Arbeit mit PPIs erweitert werden kann. Dazu wurden im zweiten Kapitel die theoretischen Hintergründe und der Forschungsstand biomedizinischer IR- und Visualisierungssysteme beschrieben. Im dritten Kapitel wurden zwei für den Prototypen zentrale Technologien vorgestellt, RDF und CYTOSCAPE WEB. Zuletzt wurden im vierten Kapitel der Aufbau des Prototyps erläutert, der im Rahmen der Masterarbeit erstellt wurde.

Die Erweiterung SEMEDICOS, zur Speicherung, Abfrage und Visualisierung von PPIs, erhöht dessen Wert für die biomedizinische Forschung. Anders als Suchmaschinen, wie FACTA+, bietet es eine Visualisierung der PPIs, wie sie die Forscher aus ihrer Arbeit mit Spezialdatenbanken und CYTOSCAPE gewohnt sind. Anders als PPI-Datenbanken, wie INTACT, wird SEMEDICO, nach Abschluss der Arbeiten an seiner Pipeline, MEDLINE komplett abdecken. Diese Kombination der Stärken verschiedener Ansätze eröffnet mehrere Weiterentwicklungsmöglichkeiten: Kurzfristig relevant sind insbesondere ein Stresstest und die Durchführung einer Nutzbarkeitsstudie zur Benutzeroberfläche. Langfristig relevant sind Überlegungen zur Nutzung aller von RDF gebotenen Optionen und zum Export der Daten.

Ein Stresstest ist nötig, da das System bisher nur einen relativ kleinen echten Datensatz speichern, abfragen und visualisieren musste. Zwar wurden Tests mit großen RDF Graphen durchgeführt, doch diese basierten immer auf

künstlichen Daten. Beim Umstieg auf die echten MEDLINE PPIs könnten sich insbesondere Probleme bei der Abfragegeschwindigkeit ergeben. Diese beträgt für den aktuellen Datensatz etwa 60ms pro Anfrage an den Triplestore. Bei der Abfrage eines Knotens mit seinem direkten Umfeld ist dies völlig ausreichend, wenn auch indirekt verbundene Knoten abgefragt werden sollen, kommt es allerdings bereits jetzt zu merklichen Verzögerungen, da für jeden direkt verbundenen Knoten eine weitere Anfrage nötig ist (vgl. 4.2.2). Hier könnte serverseitiges Filtern helfen, optional könnten auch eine Graphdatenbank als zweite Speicherlösung neben dem Triplestore getestet werden. Graphdatenbanken, wie NEO4J¹, sind auf die Abfrage von Verbindungen in Graphen spezialisiert. Sie eignen sich aber, anders als Triplestores, nicht zum Teilen der gespeicherten Informationen mit anderen Gruppen. Sollte sich eine Graphdatenbank als performanter erweisen, so könnte es deswegen sinnvoll sein beide Speicherlösungen zu nutzen – die Graphdatenbank für schnelle Suchen, den Triplestore für die Kollaboration. Unabhängig von der Speicherlösung kann die Visualisierung nur relativ kleine Graphen handhaben. Hier wird in jedem Fall serverseitige Filterung nötig sein, etwa über die Konfidenz der PPI Extraktion (vgl. 2.6). Dafür sind nur minimale programmatische Änderungen nötig, die Herausforderung besteht in der Suche nach geeigneten Schwellenwerten und der Gewichtung der einzelnen Faktoren. Eine Nutzbarkeitsstudie wäre wünschenswert, um festzustellen, wie die Benutzer mit der PPI Suche interagieren wollen. Sinnvoll dürfte in jeden Fall eine weitere optische Angleichung der NETWORKVIEWER Seite an das restliche SEMEDICO sein. Problematisch ist insbesondere die Erstellung der Anfragen. Die aktuelle Lösung kann klassische SEMEDICO Suchen oder die Suche nach PPIs handhaben, aber nicht beides gleichzeitig. Der PARSE besitzt bereits die dafür nötigen Methoden, allerdings werden diese vom QUERYTRANSLATIONSERVICE noch nicht genutzt. Neben dieser technischen Frage muss auch die Entdeckbarkeit der neuen Funktion sichergestellt werden. Hier wäre eine Eingabemaske die einfachste Lösung, in das sonstige SEMEDICO würde sich allerdings ein weiterer Ausbau der Autoergänzung der Suchleiste besser einfügen – diese könnte dem Benutzer nach Eingabe eines Proteins die

¹neo4j.org/

bekannten Interaktionsarten vorschlagen. Ein völlig neuartiger Ansatz wäre ein grafischer Editor für den Parsebaum der Eingabe. Dieser könnte als JS Programm auf dem Rechner des Benutzers ausgeführt werden und könnte die Abhängigkeiten innerhalb einer Anfrage besser verdeutlichen als Klammern. Neben diesen akuten Fragen, die spätestens während einer Betaphase geklärt werden sollten, bietet RDF eine Reihe an Möglichkeiten, die bisher ungenutzt bleiben (vgl. Abschnitt 3.1). Einerseits ist RDF grundsätzlich zum Austausch und zur Verbindung verschiedener Datensätze gedacht. Dieses Ziel der Linked Open Data könnte auch von SEMEDICO umgesetzt werden. Dazu würde man den RDF Graph der PPIs, über in VIRTUOSO bereits vorhandene Funktionen, öffentlich zugänglich machen. Dies sollte für die Eindeutigkeit der URIs sorgen, über die die PPIs von außen erreichbar wären. Gelöst werden muss noch die Permanenz der URIs. Diese sollten sich möglichst nie ändern, da ungültige Links die Nutzbarkeit des Systems bedrohen [Berners-Lee 1998; Stein 2003]. Zudem könnte das RDF auch genutzt werden, um, in Verbindung mit einer in OWL erstellten Ontologie, automatisch Schlussfolgerung zu ziehen oder Hypothesen zu entwickeln (vgl. 2.1).

Eine letzte mögliche Weiterentwicklung wäre der Export der Daten. Bisher wird vom Server zwar XGMML erstellt, das von CYTOSCAPE gelesen werden kann, es gibt aber noch keine Möglichkeit dieses herunterzuladen. Gerade bei größeren Netzwerken, die CYTOSCAPE WEB nur schwer anzeigen kann, wäre dies eine interessante Option für die Benutzer. Dabei könnte ein Umstieg vom bisher generierten XGMML auf ein anderes Format, wie PSI MI² (Proteomics Standards Initiative Molecular Interaction XML Format) sinnvoll sein. Anders als in XGMML, können darin standardisierte Informationen zu den durchgeführten Experimenten gespeichert werden, weswegen die Inhalte vieler Interaktionsdatenbanken auch in PSI MI zum Download angeboten werden [Kerrien u. a. 2012; Ceol u. a. 2010]. Allerdings werden diese Informationen bisher nicht aus den Texten extrahiert, das PSI MI Format könnte also nur mit vielen Leerstellen eingehalten werden. Hier wäre eine Befragung der Nutzer sinnvoll, ob das Format ihnen dennoch Vorteile bieten würde.

²psidev.sourceforge.net/mi/xml/doc/user/

Diese Übersicht macht klar, wie der Prototyp alltagstauglich gemacht werden kann, damit er der biomedizinischen Forschung hoffentlich schon bald zur Verfügung steht.

Literaturverzeichnis

Adar 2006

ADAR, Eytan: GUESS: a language and interface for graph exploration. In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*, 2006 (CHI '06), S. 791–800

Altman u. a. 2008

ALTMAN, Russ ; BERGMAN, Casey ; BLAKE, Judith ; BLASCHKE, Christian ; COHEN, Aaron ; GANNON, Frank ; GRIVELL, Les ; HAHN, Udo ; HERSH, William ; HIRSCHMAN, Lynette ; JENSEN, Lars ; KRALLINGER, Martin ; MONS, Barend ; O'DONOGHUE, Sean ; PEITSCH, Manuel ; REBHOLZ-SCHUHMANN, Dietrich ; SHATKAY, Hagit ; VALENCIA, Alfonso: Text mining for biology - the way forward: opinions from leading scientists. In: *Genome Biology* 9 (2008), Nr. Suppl 2, S. S7

Bastian u. a. 2009

BASTIAN, Mathieu ; HEYMANN, Sebastien ; JACOMY, Mathieu: Gephi: An Open Source Software for Exploring and Manipulating Networks. In: *International AAAI Conference on Weblogs and Social Media*, 2009

Berg u. a. 2007

BERG, Jeremy M. ; TYMOCZKO, John L. ; STRYER, Lubert: *Stryer Biochemie*. 6. Auflage. 2007

Berners-Lee 1998

BERNERS-LEE, Tim: *Cool URIs don't change*. www.w3.org/Provider/Style/URI. Version: 1998

Bizer u. a. 2009

BIZER, Christian ; HEATH, Tom ; BERNERS-LEE, Tim: Linked Data – The Story So Far. In: *International Journal on Semantic Web and Information Systems* 5 (2009), Nr. 3, S. 1–22

Brachman u. Levesque 2004

BRACHMAN, Ronald J. ; LEVESQUE, Hector J.: *Knowledge representation and reasoning*. 2004

Buyko 2012

BUYKO, Ekaterina: *Event Extraction from Biomedical Texts using Trimmed Dependency Graphs*, Friedrich-Schiller-Universität Jena, Diss., 2012

Buyko u. a. 2009

BUYKO, Ekaterina ; FAESSLER, Erik ; WERMTER, Joachim ; HAHN, Udo: Event Extraction from Trimmed Dependency Graphs. In: *Proceedings of Natural Language Processing in Biomedicine (BioNLP) NAACL 2009 Workshop*, 2009, S. 19–27

Buyko u. a. 2011

BUYKO, Ekaterina ; FAESSLER, Erik ; WERMTER, Joachim ; HAHN, Udo: Syntactic Simplification and Semantic Enrichment - Trimming Dependency Graphs for Event Extraction. In: *Computational Intelligence* 27 (2011), Nr. 4

Cannataro u. a. 2010

CANNATARO, Mario ; GUZZI, Pietro H. ; VELTRI, Pierangelo: Using RDF for managing protein-protein interaction data. In: *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, 2010 (BCB '10), S. 664–670

Ceol u. a. 2010

CEOL, Arnaud ; CHATR-ARYAMONTRI, Andrew ; LICATA, Luana ; PELUSO, Daniele ; BRIGANTI, Leonardo ; PERFETTO, Livia ; CASTAGNOLI, Luisa ; CESARENI, Gianni: MINT, the molecular interaction database: 2009 update. In: *Nucleic Acids Research* 38 (2010), Nr. Database-Issue, S. 532–539

Cormen u. a. 2001

CORMEN, T.H. ; LEISERSON, C.E. ; RIVEST, R.L. ; STEIN, C.: *Introduction to Algorithms*. 2. Auflage. 2001

De Las Rivas u. Fontanillo 2010

DE LAS RIVAS, J. ; FONTANILLO, C.: Protein–Protein Interactions Essentials: Key Concepts to Building and Analyzing Interactome Networks. In: *PLoS Computational Biology* 6 (2010), Nr. 6, S. e1000807

Decker u. a. 2000

DECKER, Stefan ; MELNIK, Sergey ; HARMELEN, Frank V. ; FENSEL,

- Dieter ; KLEIN, Michel ; BROEKSTRA, Jeen ; ERDMANN, Michael ; HORROCKS, Ian: The Semantic Web: The Roles of XML and RDF. In: *IEEE Internet Computing* 4 (2000), S. 63–74
- Doms u. Schroeder 2005**
DOMS, Andreas ; SCHROEDER, Michael: GoPubMed: exploring PubMed with the Gene Ontology. In: *Nucleic Acids Research* 33(Web Server issue) (2005), S. W783–W786
- Duan u. a. 2011**
DUAN, Songyun ; KEMENTSIETSIDIS, Anastasios ; SRINIVAS, Kavitha ; UDREA, Octavian: Apples and oranges: a comparison of RDF benchmarks and real RDF datasets. In: *Proceedings of the 2011 international conference on Management of data*, 2011 (SIGMOD’11), S. 145–156
- Faessler 2009**
FAESSLER, Erik: *Automatische Extraktion von Protein-Protein-Interaktionen in biomedizinischen Texten unter Verwendung von Supportvektormaschinen*, Friedrich-Schiller-Universität Jena, Diplomarbeit, 2009
- Faessler u. a. 2009**
FAESSLER, Erik ; LANDEFELD, Rico ; TOMANEK, Katrin ; HAHN, Udo: LuCas - A Lucene CAS Indexer. In: *Proceedings of the Biennial GSCL Conference*, 2009
- Fluit u. a. 2002**
FLUIT, Christiaan ; SABOU, Marta ; HARMELEN, Frank van: Ontology-based Information Visualisation: Towards Semantic Web Applications. In: GEROIMENKO, Vladimir (Hrsg.): *Visualising the Semantic Web*. 2002, S. 36–48
- Friedman u. Johnson 2006**
FRIEDMAN, Carol ; JOHNSON, Stephen B.: Natural Language and Text Processing in Biomedicine. In: SHORTLIFFE, Edward H. (Hrsg.) ; CIMINO, James J. (Hrsg.): *Biomedical Informatics*. 3. Auflage. 2006 (Health Informatics), Kapitel 8
- Ge u. a. 2003**
GE, Hui ; WALHOUT, Albertha J. ; VIDAL, Marc: Integrating ‘omic’ information: a bridge between genomics and systems biology. In: *Trends in Genetics* 19 (2003), Nr. 10, S. 551–560

Hahn u. a. 2007a

HAHN, U. ; WERMTER, J. ; BLASCZYK, R. ; HORN, P.: Text Mining: Powering the Database Revolution. In: *Nature* 448 (2007), Nr. 7150, S. 130

Hahn u. a. 2007b

HAHN, Udo ; BUYKO, Ekaterina ; TOMANEK, Katrin ; PIAO, Scott ; MCNAUGHT, John ; TSURUOKA, Yoshimasa ; ANANIADOU, Sophia: An annotation type system for a data-driven NLP pipeline. In: *The LAW at ACL 2007 – Proceedings of the Linguistic Annotation Workshop*, 2007, S. 33–40

Hahn u. a. 2007c

HAHN, Udo ; WERMTER, Joachim ; DELUCA, David S. ; BLASCZYK, Rainer ; POPRAT, Michael ; BAJWA, Asad ; HORN, Peter A.: Stemnet – An Evolving Service for Knowledge Networking in the Life Sciences. In: *GES 2007 – Proceedings of the German e-Science Conference*, 2007

Hersh 2003

HERSH, William R.: *Information Retrieval: A Health and Biomedical Perspective*. 2. Auflage. 2003 (Health Informatics)

Horrocks u. a. 2003

HORROCKS, Ian ; PATEL-SCHNEIDER, Peter F. ; HARMELEN, Frank van: From SHIQ and RDF to OWL: The Making of a Web Ontology Language. In: *Journal of Web Semantics* 1 (2003), Nr. 1, S. 7–26

Iragne u. a. 2005

IRAGNE, Florian ; NIKOLSKI, Macha ; MATHIEU, Bertrand ; AUBER, David ; SHERMAN, David: ProViz: protein interaction visualization and exploration. In: *Bioinformatics* 21 (2005), Nr. 2, S. 272–274

Jurafsky u. Martin 2009

JURAFSKY, Daniel ; MARTIN, James H.: *Speech and language processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2. Auflage. 2009 (Prentice Hall Series in Artificial Intelligence)

Kerrien u. a. 2012

KERRIEN, Samuel ; ARANDA, Bruno ; BREUZA, Lionel ; BRIDGE, Alan ; BROACKES-CARTER, Fiona ; CHEN, Carol ; DUESBURY, Margaret ; DUMOUSSEAU, Marine ; FEUERMANN, Marc ; HINZ, Ursula ; JANDRASITS, Christine ; JIMENEZ, Rafael C. ; KHADAKE, Jyoti ; MAHADEVAN, Usha

- ; MASSON, Patrick ; PEDRUZZI, Ivo ; PFEIFFENBERGER, Eric ; PORRAS, Pablo ; RAGHUNATH, Arathi ; ROECHERT, Bernd ; ORCHARD, Sandra ; HERMJAKOB, Henning: The IntAct molecular interaction database in 2012. In: *Nucleic Acids Research* 40 (2012), Nr. D1
- Knecht u. Nelson 2002**
KNECHT, Lou Wave S. ; NELSON, Stuart J.: Mapping in PubMed. In: *Journal of the Medical Library Association* 90 (2002), Nr. 4, S. 475
- Lacy 2005**
LACY, Lee W.: *Owl: Representing Information Using the Web Ontology Language*. 2005
- Lopes u. a. 2010**
LOPES, Christian T. ; FRANZ, Max ; KAZI, Farzana ; DONALDSON, Sylvia L. ; MORRIS, Quaid ; BADER, Gary D.: Cytoscape Web: an interactive web-based network browser. In: *Bioinformatics* (2010)
- Lu 2011**
LU, Zhiyong: PubMed and beyond: a survey of web tools for searching biomedical literature. In: *Database* 2011 (2011)
- Michel u. a. 2011**
MICHEL, Jean-Baptiste ; SHEN, Yuan K. ; AIDEN, Aviva P. ; VERES, Adrian ; GRAY, Matthew K. ; THE GOOGLE BOOKS TEAM ; PICKETT, Joseph P. ; HOIBERG, Dale ; CLANCY, Dan ; NORVIG, Peter ; ORWANT, Jon ; PINKER, Steven ; NOWAK, Martin A. ; AIDEN, Erez L.: Quantitative Analysis of Culture Using Millions of Digitized Books. In: *Science* 331 (2011), Nr. 6014, S. 176–182
- Milian u. a. 2009**
MILIAN, Krystyna ; ALEKSOVSKI, Zharko ; VDOVJAK, Richard ; TEIJE, Annette ten ; HARMELEN, Frank van: Identifying disease-centric subdomains in very large medical ontologies. In: *Proceedings of the AIME'09 workshop on Knowledge Representation for Healthcare*, 2009, S. 41–50
- Müller u. a. 2003**
MÜLLER, Hans-Michael ; KENNY, Eimear E. ; STERNBERG, Paul W.: Textpresso: an ontology-based information retrieval and extraction system for biological literature. In: *PLoS Biology* 2 (2003), Nr. 11
- Newman u. a. 2008**
NEWMAN, Andrew ; HUNTER, Jane ; LI, Yuan-Fang ; BOUTON, Chris

- ; DAVIS, Melissa: BioMANTA Ontology: The integration of protein-protein interaction data. In: *InterOntology08*, 2008
- Ottmann u. Widmayer 2002**
 OTTMANN, T. ; WIDMAYER, P.: *Algorithmen und Datenstrukturen*. 4. Auflage. 2002
- Ralph Grishman 1996**
 RALPH GRISHMAN, Beth S.: Message Understanding Conference – 6: A Brief History. In: *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, 1996, S. 466—471
- Schneider 2007**
 SCHNEIDER, Anne: *Linguistik, Semantik und Information Retrieval – Eine semantische Suchmaschine in der Biomedizin*. www.fs-sprachwissenschaft.uni-tuebingen.de/tacos/index.php5?lang=de&content=agenda. Version: 2007
- Schneider u. a. 2009**
 SCHNEIDER, Anne ; LANDEFELD, Rico ; WERMTER, Joachim ; HAHN, Udo: Do users appreciate novel interface features for literature search?: a user study in the life sciences domain. In: *Proceedings of the 2009 IEEE international conference on Systems, Man and Cybernetics*, 2009 (SMC'09), S. 2062–2067
- Searls 2005**
 SEARLS, David B.: Data integration: challenges for drug discovery. In: *Nature Reviews Drug Discovery* 4 (2005), S. 45–58
- Shannon u. a. 2003**
 SHANNON, Paul ; MARKIEL, Andrew ; OWEN, Ozier ; BALIGA, Nitin S. ; WANG, Jonathan T. ; RAMAGE, Daniel ; AMIN, Nada ; SCHWIKOWSKI, Benno ; IDEKER, Trey: Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. In: *Genome Research* 13 (2003), S. 2498–2504
- Shneiderman 1996**
 SHNEIDERMAN, Ben: The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In: *Proceedings of the 1996 IEEE Symposium on Visual Languages*, 1996, S. 336–344
- Smiley u. Pugh 2009**
 SMILEY, David ; PUGH, Eric: *Solr 1.4 Enterprise Search Server*. 2009

Stein 2003

STEIN, Lincoln D.: Integrating biological databases. In: *Nature Reviews Genetics* 4 (2003), S. 337–345

Stevens u. a. 2000

STEVENS, Robert ; BAKER, Patricia ; BECHHOFFER, Sean ; NG, Gary ; JACOBY, Alex ; PATON, Norman W. ; GOBLE, Carole A. ; BRASS, Andy: TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. In: *Bioinformatics* 16 (2000), Nr. 2, S. 184–186

Stumpf u. a. 2008

STUMPF, Michael P. H. ; THORNE, Thomas ; SILVA, Eric de ; STEWART, Ronald ; AN, Hyeong J. ; LAPPE, Michael ; WIUF, Carsten: Estimating the size of the human interactome. In: *PNAS* 105 (2008), Nr. 19, S. 6959–6964

Tsuruoka u. a. 2011

TSURUOKA, Yoshimasa ; MIWA, Makoto ; HAMAMOTO, Kaisei ; TSUJII, Jun'ichi ; ANANIADOU, Sophia: Discovering and visualizing indirect associations between biomedical concepts. In: *Bioinformatics* Bd. 27, 2011, S. i111–i119

Tsuruoka u. a. 2008

TSURUOKA, Yoshimasa ; TSUJII, Jun'ichi ; ANANIADOU, Sophia: FACTA: a text search engine for finding associated biomedical concepts. In: *Bioinformatics* 24 (2008), Nr. 21, S. 2559–2560

UIMA

UIMA. *Overview & Sdk Setup*. www.uima.apache.org/d/uimaj-2.3.1/overview_and_setup.pdf

Urbani u. a. 2010

URBANI, Jacopo ; KOTOULAS, Spyros ; MAASSEN, Jason ; HARMELEN, Frank van ; BAL, Henri: OWL reasoning with WebPIE: calculating the closure of 100 billion triples. In: *Proceedings of the Seventh European Semantic Web Conference* Bd. 6088, 2010 (LNCS), S. 213–227

Uschold u. Gruninger 2004

USCHOLD, Michael ; GRUNINGER, Michael: Ontologies and semantics for seamless connectivity. In: *SIGMOD Rec.* 33 (2004), S. 58–64

Vrandecic 2010

VRANDECIC, Denny: *Ontology Evaluation*. Karlsruhe, KIT, Fakultät für Wirtschaftswissenschaften, Diss., 2010

Wu u. a. 2006

WU, Cathy H. ; APWEILER, Rolf ; BAIROCH, Amos ; NATALE, Darren A. ; BARKER, Winona C. ; BOECKMANN, Brigitte ; FERRO, Serenella ; GASTEIGER, Elisabeth ; HUANG, Hongzhan ; LOPEZ, Rodrigo ; MAGRANE, Michele ; MARTIN, Maria J. ; MAZUMDER, Raja ; O'DONOVAN, Claire ; REDASCHI, Nicole ; SUZEK, Baris E.: The Universal Protein Resource (UniProt): an expanding universe of protein information. In: *Nucleic Acids Research* 34 (2006), Nr. Database-Issue, S. 187–191

Xenarios u. a. 2000

XENARIOS, Ioannis ; RICE, Danny W. ; SALWINSKI, Lukasz ; BARON, Marisa K. ; MARCOTTE, Edward M. ; EISENBERGA, David: DIP: the Database of Interacting Proteins. In: *Nucleic Acids Research* 28 (2000), Nr. 1, S. 289–291

Abbildungsverzeichnis

1.1	MEDLINE Einträge	2
2.1	Beispiel für einen Graphen	6
2.2	Artikel und Experimente in INTACT	9
2.3	Ergebnisseite des aktuellen SEMEDICOS	13
2.4	Aufbau und Funktion SEMEDICOS	14
2.5	Beispiel für eine computerlinguistische Pipeline	14
2.6	Computerlinguistische Pipeline mit JREX	17
2.7	Biomedizinische Beispielsgrafik aus Berg u. a. [2007]	18
2.8	Interaktionsnetzwerk in CYTOSCAPE	20
2.9	Filterung biomedizinischer Netzwerke in MINT	22
3.1	Verhältnis Artikel zu PPIs	30
3.2	CYTOSCAPE WEB Darstellungsoptionen	33
4.1	Aufbau des erweiterten SEMEDICOS	37
4.2	Beispiel für einen Parsebaum	44
4.3	Beispiel für einen PPI Graphen	46
4.4	SEMEDICO Ergebnisseite mit PPI Link	47
4.5	Ergebnisseite des aktuellen SEMEDICOS	48
4.6	PPI Visualisierung im Übersichtsmodus	49
4.7	PPI Visualisierung mit Details	50
4.8	Interaktion mit dem Prototyp	51

Eigenständigkeitserklärung

Ich erkläre, dass ich vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Hilfsmittel und Quellen angefertigt habe.

Die eingereichte Arbeit ist nicht anderweitig als Prüfungsleistung verwendet worden oder in deutscher oder einer anderen Sprache als Veröffentlichung erschienen.

Seitens des Verfassers bestehen keine Einwände, die vorliegende Masterarbeit für die öffentliche Benutzung zur Verfügung zu stellen.

Jena, 28. März 2012

Johannes Hellrich