Dependenz-basierte Relationsextraktion mit der UIMA-basierten Textmining-Pipeline UTEMPL

Jannik Strötgen[1,2], Juliane Fluck[1] und Anke Holler[3]

[1] Fraunhofer Institut für Algorithmen und wissenschaftliches Rechnen SCAI
juliane.fluck@scai.fraunhofer.de
[2] Institut für Informatik, Ruprecht-Karls-Universität Heidelberg
jannik.stroetgen@informatik.uni-heidelberg.de
[3] Seminar für Deutsche Philologie, Georg-August-Universität Göttingen
anke.holler@phil.uni-goettingen.de

Zusammenfassung. Der Artikel beschreibt das UIMA-basierte Textmining-System UTEMPL, das für die spezifischen Anforderungen der Verarbeitung biomedizinischer Fachliteratur entwickelt wurde. Anhand der sog. Protein-Protein-Interaktionen wird dargestellt, wie dieses flexible und modular aufgebaute System zur Relationsextraktion genutzt werden kann. Eine Evaluierung anhand verschiedener Korpora zeigt, dass ein linguistisch motivierter, dependenzbasierter Ansatz zur Relationsextraktion in seiner Leistungsfähigkeit einem einfachen Pattern-Matching-Ansatz meist überlegen ist.

1 Einleitung

In aktiven Forschungsbereichen wie der Biomedizin liegen neue Erkenntnisse vor allem als unstrukturierte Textdaten vor. Zugleich wächst die Zahl der wissenschaftlichen Publikationen exponentiell [15]. Eine rein stichwortbasierte Suche ist kaum noch ausreichend, um in den vorhandenen großen Beständen biomedizinischer Fachliteratur neu gewonnenes Wissen zu identifizieren. Zugleich stellt die Biomedizin domänenspezifische Anforderungen, da in den Dokumenten sowohl nicht-eindeutige Gen- und Proteinnamen als auch domänenspezifische Relationen zwischen verschiedenen Namensentitäten, wie z.B. Protein-Protein-Interaktionen (PPIs), erkannt werden müssen. Vor diesem Hintergrund gewinnt der Einsatz leistungsfähiger Textmining (TM)-Verfahren in der biomedizinischen Forschung zunehmend an Bedeutung.

Im vorliegenden Artikel beschreiben wir eine modular aufgebaute, flexible Softwareumgebung, die die Kombination verschiedener TM-Komponenten erlaubt, und zeigen, wie diese zur Extraktion von PPIs aus fachwissenschaftlichen biomedizinischen Texten eingesetzt werden kann. Der Artikel ist folgendermaßen strukturiert: Im nachfolgenden Abschn. 2 stellen wir die für die biomedizinische Domäne entwickelte TM-Pipeline UTEMPL vor. Dabei gehen wir auf die Architektur UIMA und auf die Anforderungen der biomedizinischen Domäne ein. Abschn. 3 widmet sich Methoden der Relationsextraktion und Abschn. 4 präsentiert einen neuen, in UTEMPL realisierten Ansatz zur Extraktion von PPIs. Abschn. 5 diskutiert die auf verschiedenen Korpora erzielten Evaluationsergebnisse.

2 Die Textmining-Pipeline UTEMPL

Das UIMA-basierte System UTEMPL wurde mit dem Ziel entwickelt, TM-Aufgaben mit wechselnden Anforderungen im Bereich der biomedizinischen Domäne zu lösen. Ein klarer Vorzug der erstellten TM-Pipeline liegt in ihrem modularen Aufbau, so dass einzelne Komponenten unaufwändig angepasst bzw. ausgetauscht werden können, falls (i) neue, bessere Komponenten zur Verfügung stehen oder (ii) die Anforderungen sich geändert haben.

2.1 UIMA

UTEMPL basiert auf der frei verfügbaren, plattformunabhängigen Architektur und Software-Umgebung UIMA (*Unstructured Information Management Architecture*). Diese ermöglicht es, unstrukturierte Daten verschiedener Art (Text, Audiodaten, Bilder) zu verarbeiten[4], und erlaubt zudem, verschiedene Komponenten und Suchtechnologien derart zu verknüpfen, dass eine Pipeline von interagierenden Tools entsteht. Dadurch, dass alle Komponenten der Pipeline auf eine gemeinsame Datenstruktur zugreifen, die sog. Common Analysis Structure (CAS), können auch Werkzeuge verbunden werden, die zunächst nicht für ein Zusammenspiel entwickelt wurden.

Eine UIMA-Pipeline besteht aus Komponenten der folgenden drei Arten: mindestens einem Collection Reader, einer Analysis Engine und einem CAS Consumer. Der *Collection Reader* (CR) liest Daten, wie Textdokumente, von einer Datenquelle (Datenbank, Filesystem etc.) ein. Durch den CR wird festgelegt, wie über die einzelnen Dokumente iteriert wird und welche Teile der Dokumente weiter analysiert werden sollen. Der CR erstellt zu diesem Zweck für jedes Input-Dokument ein CAS-Objekt, das neben dem in unserem Fall wichtigen Dokumententext zusätzlich Metadaten enthalten kann. Diese CAS-Objekte werden an die Analysis Engines weitergereicht. Die *Analysis Engines* (AEs) sind die Bestandteile der UIMA-Pipeline, die das jeweilige Dokument analysieren, Informationen finden und annotieren. Die erste AE einer Pipeline bekommt das CAS-Objekt direkt vom CR, spätere AEs von der jeweils vorigen. Die von AEs gefundenen oder abgeleiteten Informationen, die sog. Analysis Results, beinhalten typischerweise Metainformationen über den Inhalt des Dokuments. AEs können sowohl auf den Dokumententext innerhalb des CAS-Objektes zugreifen als auch auf die Analysis Results voriger AEs[5]. Der *CAS Consumer* ist das letzte Glied innerhalb der UIMA Pipeline und führt die abschließende Verarbeitung des CAS-Objektes durch. Anders als ein CR oder eine AE fügt der CAS Consumer dem CAS-Objekt keine weiteren Metainformationen hinzu. Typische Aufgaben eines CAS Consumers sind stattdessen, relevante Elemente aus dem CAS-Objekt zu extrahieren oder zu visualisieren, einen Suchindex für den Inhalt der CAS-Objekte aufzubauen oder mit Hilfe eines Goldstandards eine Evaluierung vorzunehmen.

[4] UIMA wurde von IBM entwickelt (`http://www.research.ibm.com/UIMA/`); seit 2006 wird die Entwicklung bei der Apache Software Foundation (`http://incubator.apache.org/uima/`) als Open-Source-Projekt fortgeführt.

[5] Bspw. Informationen über die Sätze eines Dokuments mit Positionsangaben.

2.2 Anforderungen an die Verarbeitung biomedizinischer Texte

Die Entwicklung von TM-Komponenten für die Extraktion von biomedizinischer Information muss der Tatsache Rechnung tragen, dass die biomedizinische Sprache eine Subsprache darstellt[6] und daher durch Besonderheiten im Bereich der Lexik und der Syntax gekennzeichnet ist [5]. So enthält das biomedizinische Vokabular vor allem nominale Ausdrücke, wie Protein- und Gennamen oder Bezeichnungen für biomedizinische Verfahren, sowie verbale Lexeme zur Beschreibung relationaler Beziehungen, wie z.B. *to inhibit*, *to phosphorylate* oder *to bind* [1]. Hinzu kommen terminologische Besonderheiten, wodurch Lexeme eine andere Bedeutung aufweisen als in der Standardsprache. Bspw. deutet das Verb *to associate* in der biomedizinischen Subsprache i.d.R. auf eine Interaktion im Sinne von *binding* hin [3]. Im Bereich der Syntax ist die biomedizinische Subsprache zum Einen durch häufig auftretende Passivierungen geprägt, die als verbale (*A was activated by B*), als adjektivische (*B-activated A*) und als nominale (*A activation by B*) Formen vorkommen. Zum Anderen lassen sich vor allem syntaktische Muster der Form *Protein-Interaktionsverb-Protein* beobachten.

Aus den genannten Eigenschaften der biomedizinischen Subsprache ergeben sich spezifische Probleme bei der Verarbeitung biomedizinischer Textbestände, insbesondere weil gängige NLP-Werkzeuge vorrangig für die Verarbeitung standardsprachlicher Dokumente entwickelt und bzgl. standardsprachlicher Korpora evaluiert wurden. Beispielsweise werden überdurchschnittlich viele Lexeme bei der Verarbeitung biomedizinischer Texte nicht erkannt. Dies betrifft *named entities* wie Gen- und Proteinnamen oder Verben, die für die Domäne typische Interaktionen beschreiben (*downregulate*, *upregulate* etc.). Wie [14], [8] und [1] unabhängig voneinander zeigen, können nur domänenspezifische Anpassungen der jeweiligen NLP-Werkzeuge zu besseren Verarbeitungsergebnissen führen. Die Pipeline UTEMPL, die im nächsten Abschn. beschrieben wird, ermöglicht es, solche Anpassungen mit angemessenem Aufwand vorzunehmen.

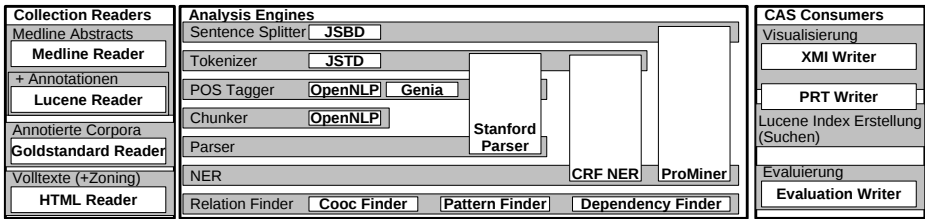

Abb. 1. Relevante, in UTEMPL integrierte Komponenten mit ihren Aufgaben.

[6] Unter einer Subsprache versteht man eine besondere, spezialisierte Form einer natürlichen Sprache, die in einer bestimmten Domäne oder einem bestimmten Fachgebiet verwendet wird.

2.3 Der Aufbau von UTEMPL

Die UIMA-basierte Pipeline UTEMPL, deren Komponenten in Abb.1 dargestellt sind, verknüpft verschiedene existierende TM-Werkzeuge in integrativer Weise, um den Belangen der biomedizinischen Domäne gerecht zu werden. Insbesondere können verschiedene Arten von Korpora von UTEMPL verarbeitet werden. UTEMPL nutzt die im Abschn. 2.1 beschriebenen Komponenten der UIMA-Architektur. Als Collection Reader stehen in UTEMPL mehrere Komponenten zur Verfügung, die jeweils unterschiedliche Formate der Eingabetexte verarbeiten. Beispielsweise wurde ein CR für Medline-Abstracts[7] eingebunden.[8] Zusätzlich sind CRs für Volltexte und Goldstandard-Korpora implementiert. So ist gewährleistet, dass alle in UTEMPL verknüpften Werkzeuge nach dem Einlesen der Daten unabhängig von der Korpusquelle verwendet werden können.

Alle Anwendungen, die die für die Relationsextraktion nötige Vorverarbeitung der Texte übernehmen, sind als Analysis Engines eingebunden. Dazu zählen Komponenten zur Satzgrenzenerkennung, zur Tokenisierung, zum Parsing und zur Named Entity Recognition. Auf Grund der UIMA-Architektur können in UTEMPL bereits existierende, speziell für die biomedizinische Domäne entwickelte Werkzeuge problemlos genutzt werden.[9] Ebenfalls als AEs sind die Komponenten zur Relationsextraktion realisiert. Wie im folgenden Abschn. genauer dargestellt wird, wurden zwei Ansätze, ein Pattern-Matching-Ansatz und ein dependenz-basierter Ansatz, umgesetzt und vergleichend evaluiert.

Für die Verarbeitung der Analyseergebnisse enthält UTEMPL verschiedene CAS Consumer. Bspw. sind CAS Consumer für die Visualisierung, für den Aufbau eines Lucene-Index sowie für die Evaluierung realisiert. Ein Vorzug der beschriebenen UIMA-basierten Pipeline UTEMPL ist, dass die Evaluierung vollständig innerhalb der Pipeline vollzogen werden kann, wenn der Collection Reader für den Goldstandard sowie der CAS Consumer für die Evaluation gemeinsam verwendet werden.

3 Relationsextraktion

In diesem Abschnitt werden zunächst drei methodische Ansätze zur Extraktion von Relationen in unstrukturierten Daten eingeführt. Danach werden zwei in UTEMPL integrierte Komponenten zur Extraktion von PPIs diskutiert.

[7] Medline ist eine Datenbank für biomedizinische Publikationen mit über 18 Millionen Eintragungen, die neben Informationen zum Autor oder zur Zeitschrift zumeist das Abstract einer Publikation enthalten. Medline ist frei über das Interface PubMed (`http://www.ncbi.nlm.nih.gov/pubmed/`) zugänglich. Da Medline eine der Hauptquellen für die Recherche im biomedizinischen Bereich ist, steigt neben der Zahl an Eintragungen auch die Zahl der Suchanfragen rapide an.

[8] Dieses Werkzeug wurde am Language and Information Engineering Lab der Universität Jena (`http://www.julielab.de`) entwickelt.

[9] Bspw. ist der Julie Sentence Boundary Detector [13] frei wählbar, und für die Namenserkennung stehen die am Fraunhofer Institut SCAI entwickelten NER-Werkzeuge, wie z.B. der ProMiner [6], zur Verfügung.

3.1 Existierende methodische Ansätze

Die einfachste Methode zur Relationsextraktion stellen auf Kookkurrenz beruhende Ansätze dar, bei denen innerhalb eines gewählten Fensters (z.B. Phrasen, Sätze, Satzpaare, Abschnitte bis hin zu vollständigen Dokumenten) gemeinsam auftretende Entitäten erfasst werden [2]. Diesem Vorgehen liegt die Hypothese zugrunde, dass gemeinsam vorkommende Entitäten in irgendeiner Weise miteinander in Beziehung stehen. Ein Vorteil von Kookkurrenzansätzen ist, dass sie sehr effizient sind, da durch ihre Einfachheit nahezu keine Vorverarbeitung der Texte nötig ist. Zudem wird durch diese Ansätze ein hoher Recall erreicht, jedoch führt die fehlende Analyse der syntaktischen und/oder semantischen Beziehungen zu einer geringen Precision.

Eine zweite gängige Methode zur Bestimmung von Relationen zwischen Entitäten ist das Pattern-Matching. Bei diesem Verfahren wird zumeist regelbasiert mit Hilfe regulärer Ausdrücke in Texten nach definierten Mustern gesucht, wobei diese Muster so spezifisch wie nötig und so allgemein wie möglich sein sollten. Dies ist nicht immer leicht umzusetzen, zumal einerseits desto mehr Muster benötigt werden, je näher diese an der syntaktischen Variation des Textes bleiben [10], andererseits aber kleine Regelsets zur Beschreibung der Muster leichter zu pflegen sind als große [11]. Im Vergleich zu Kookkurrenzansätzen ist das Pattern Matching bei der Verarbeitung nur geringfügig zeitintensiver. Allerdings sind die Definition der Muster und die Regelerstellung zeit- und arbeitsaufwändig, setzen i.d.R. Domänenwissen voraus und müssen für neue Domänen jeweils angepasst werden. Pattern-Matching-Ansätze zielen auf eine Erhöhung der Precision, gehen daher aber oft mit einer Verschlechterung des Recalls einher.

Eine dritte Methode zur Relationsextraktion stellen Ansätze dar, die auf einer tiefen linguistischen Analyse beruhen, was einen erhöhten, auch zeitlichen Aufwand für die Vorverarbeitung der Dokumente erfordert, da die Texte syntaktisch analysiert werden müssen. Dies kann durch flaches (oberflächliches) oder tiefes (vollständiges) Parsing geschehen. Relationen zwischen einzelnen Einheiten im Satz werden durch Regeln beschrieben, die auf die syntaktische Struktur rekurrieren. Dieses methodische Vorgehen zeichnet sich durch gute Recall- und Precision-Ergebnisse aus [4].

3.2 Relationsextraktion mit UTEMPL

In UTEMPL sind zwei selbst entwickelte Komponenten zur Extraktion von PPIs integriert worden: der sog. Pattern Relation Finder und der sog. Dependency Relation Finder. Während die erste Komponente den Pattern-Matching-Ansatz umsetzt, aber auch in der Lage ist, einfache Kookkurrenzen auszugeben, basiert die zweite Komponente auf einer syntaktischen Analyse von Dependenzrelationen. Für beide Komponenten wurden Listen mit domänenspezifischen nominalen und verbalen Lexemen zur Beschreibung von Relationen (Interaktionen) zwischen Entitäten erstellt. Die Lexeme wurden jeweils semantischen Kategorien zugeordnet. Die erstellten Listen (VERB4INT und NOUN4INT) können in die Analysis Engines für die Relationsextraktion als Ressource geladen werden.

Die Entwicklung der Regeln für beide Ansätze erfolgte auf der Basis eines annotierten Korpus, dem AIMed Corpus[10]. Dieses Korpus wurde mittels des Fisher-Yates-Algorithmus reproduzierbar in ein Trainings- und ein Evaluierungsset unterteilt.[11] Das Trainingsset beinhaltet 1564 Sätze (= 80%) mit durchschnittlich 0,51 Interaktionen pro Satz. Das 391 Sätze umfassende Evaluierungsset enthält im Schnitt 0,64 Interaktionen pro Satz. Für beide methodischen Ansätze wurde eine Syntax entwickelt, die die Formulierung von Regeln außerhalb des Programmcodes erlaubt.[12]

Für die Umsetzung des Pattern-Matching-Ansatzes in UTEMPL wurde in Anlehnung an [11] ein Regelset erstellt, das syntaktisch alle Eigenschaften regulärer Ausdrücke aufweist. Der Pattern-Matching-Ansatz ist in UTEMPL als Analysis Engine realisiert. Im folgenden Abschn. wird detailliert erläutert, wie der dependenz-basierte Ansatz in UTEMPL realisiert wurde.

4 Dependenz-basierte Extraktion von PPIs

Der dependenz-basierte Ansatz ist inspiriert durch das RelEx-System von [4]. Die gemeinsame Grundidee ist, Regeln für die Extraktion von Protein-Protein-Interaktionen zu entwerfen, die sich auf die Ausgabe eines Dependency-Parsers stützen. Anders als RelEx, das mit drei sehr allgemeinen Regeln arbeitet, wird in UTEMPL versucht, für verschiedene syntaktische Phänomene spezifische Regeln zu formulieren. Auf diese Weise sollen zufällig gemeinsam auftretende Entitäten von miteinander interagierenden Entitäten unterschieden werden.[13] Es ist insgesamt zu erwarten, dass der dependenz-basierte Ansatz zumindest bzgl. der Precision einem einfachen Pattern Matching überlegen ist.

Das Vorgehen bei der dependenz-basierten Relationsextraktion ist folgendermaßen: Nach der Bestimmung der Satzgrenzen und der Entitäten werden zunächst alle Sätze gesucht, die eine Kookkurrenz enthalten.[14] Diese Sätze werden dann an den Stanford Parser weitergeleitet, der ein Part-of-Speech-Tagging und eine Syntaxanalyse vornimmt. Der UIMA-Wrapper des Stanford Parsers[15] wird so erweitert, dass dem Parser die durch eine NER-Anwendung gefundenen Entitäten mitgeteilt werden. Dadurch können Fehler vermieden und Verarbeitungszeit eingespart werden, da durch diese Erweiterung alle Entitäten als Eigennamen und als einzelne Token behandelt werden. Auf der Dependenzausgabe des

[10] `ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/`

[11] Ein solches Vorgehen ermöglicht die Evaluierung auf Korpora, die bei der Regelentwicklung unberücksichtigt geblieben sind.

[12] Anpassungen und Erweiterungen können daher entwicklerunabhängig ohne Änderung des Programmcodes vollzogen werden.

[13] Darüber hinaus sollte die lineare Distanz zwischen interagierenden Entitäten innerhalb eines Satzes von geringer Bedeutung sein, zumindest solange der Dependency-Parser den jeweiligen Satz weitgehend korrekt analysieren kann.

[14] So wird vermieden, dass der Dependency-Parser Sätze analysiert, in denen aufgrund fehlender Entitäten keine Relationen gefunden werden können.

[15] Dieser ist Teil des UIMA bioNLP Toolkits, `http://bionlp-uima.sourceforge.net`.

Stanford Parsers werden die Regeln zur Extraktion von Relationen (Interaktionen) angewandt. Dabei kann sowohl auf die Relationen, die der Stanford-Parser als Typen für die Verbindungen zwischen den einzelnen Lexemen ausgibt [9], als auch auf die bereits erwähnten Ressourcen VERB4INT und NOUN4INT zurückgegriffen werden. Um die einzelnen Komponenten von UTEMPL so flexibel wie möglich zu gestalten, wird der Dependency Relation Finder als eine eigene Analysis Engine entwickelt, die neben den Ressourcen für die Interaktionen ausdrückende Lexeme auf Annotationen im CAS zurückgreift. Zu diesen Annotationen gehören die Ausgaben einer NER Anwendung, des Stanford Parsers sowie des als AE entwickelten Cooccurrence Finders. Als weitere Ressource werden dem Dependency Relation Finder die erarbeiteten Regeln übergeben.

4.1 Regelsyntax

Zur Verdeutlichung der Syntaxeigenschaften der Regelsprache dient Abb. 2, in der die Dependenzausgaben des Stanford Parsers für die relevanten Bereiche zweier Beispielsätze aus dem AIMed Corpus in Baumstruktur dargestellt sind.

Es werden folgende vier Entitätenpaare betrachtet, wobei die jeweilige Tokennummer der Entitäten in Klammern angegeben ist. Aufgrund der Übergabe der Entitäten durch den Namenserkenner behandelt der Dependency-Parser auch mehrwortige Entitäten als ein Token.

A: **calnexin** (7) und **calreticulin** (9) in S1
B: **calnexin** (7) und **Glut 1** (12) in S1
C: **calreticulin** (9) und **Glut 1** (12) in S1
D: **IRS-1** (23) und **insulin receptor** (26) in S2

Im ersten Beispielsatz (S1) sollen B und C, nicht jedoch A als Interaktionen erkannt werden, in S2 soll D als Interaktion extrahiert werden. Ein wichtiges Charakteristikum ist der gemeinsame Elternknoten, der entweder eine der Entitäten ist (A, D) oder ein anderes Wort (B, C). Diese Unterscheidung führt zu zwei Regelmengen, die **e1isCP** und **otherCP** genannt werden.

Das Ziel der Syntax ist, dass mit Befehlen Bedingungen an den Dependenzbaum gestellt werden können. Die Positionen (Knoten) können von den En-

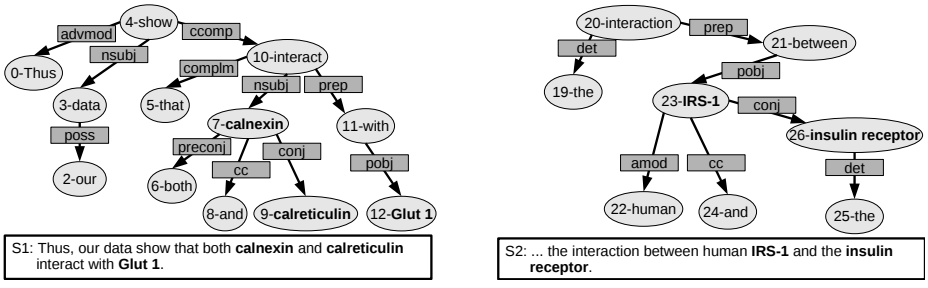

Abb. 2. Dependenzausgabe des Stanford Parsers für Beispiele aus dem AIMed Corpus (Wörter mit Tokennummern in Ellipsen; Relationen in Rechtecken; Entitäten fett).

titäten (E1, E2) und vom gemeinsamen Elternknoten (CP) aus angesprochen werden. Die Befehle sind **CP**, **e1CP-1**, **e1CP-2**, **e2CP-1** und **e2CP-2** (1 bzw. 2 unterhalb CP Richtung E1 bzw. E2) sowie **pE1**, **gpE1**, **ggpE1** und **gggpE1** (1, 2, 3 bzw. 4 oberhalb von E1). Als Attribute erhalten die Befehle einen String, ein Lexem aus INOUN4INT (**INOUN**) oder aus IVERB4INT (**IVERB**). Diese Attribute enthält auch der Befehl **anywhere**, der alle Positionen zwischen E1 bzw. E2 und CP überprüft.

Die Relationen (Pfade) zwischen den Entitäten und CP oder dem Wurzelknoten werden mit **e1RelToCP**, **e2RelToCP** und **e1RelToRoot** angesprochen und enthalten Relationsangaben (z.B. nsubj, dobj), die der Dependency-Parser ausgibt. Die Genauigkeit der Relationen kann mit **e1RelType** und **e2RelType** bestimmt werden: **equals**, **starts**, **ends** oder **contains**.

Außerdem können die maximalen Relationsentfernungen bestimmt (**e1RelMax** und **e2RelMax**), eine Negationsprüfung verlangt (**checkNeg**:yes) und eine Regel als Negation verwendet werden, wodurch nicht interagierende Entitätenpaare ausgeschlossen werden können (**isNegation**:yes).

Mit diesen Befehlen und der Berücksichtigung der Regelgruppen **otherCP** und **e1isCP** können die Regeln B1 und B2 aus Abb. 3 geschrieben werden, die für die oben genannten Relationen B, C und D zutreffen, für A richtigerweise jedoch nicht.

4.2 Regelentwicklung

Die Entwicklung der Regeln ist in einzelne Schritte aufgeteilt. Zunächst wird versucht präzise Regeln zu schreiben, die eine hohe Precision erzielen. Dadurch können ein Regelset für hohe Precision und eines für hohe F-Score-Werte entwickelt werden. In Abb. 3 sind Regeln für einige der in den Entwicklungsschritten beschriebenen Konstruktionen aufgeführt und in Abb. 4 sind die Ergebnisse der Entwicklungsschritte auf dem Trainings- und Evaluierungsset angegeben.

dep-1 Zunächst werden Regeln für einfache, aber häufig auftretende Relationen geschrieben, die teilweise in Abb. 3 aufgeführt sind. Die Beispiele sind Generalisierungen, es reicht aus, dass A und B der Kopf ihrer NP sind:

(1) *A binds B* (4) *A was activated by B* (7) *Binding of A to B*
(2) *A interacts with B* (5) *Activation of A by B* (8) *Interaction between A and B*
(3) *A binds to B* (6) *Interaction of A with B*

Dass bereits False Positives (FP) extrahiert werden (siehe Abb. 4), liegt daran, dass spekulative Interaktionen im AIMed Corpus meistens nicht annotiert sind, in UTEMPL jedoch weder ausgeschlossen noch als Negation betrachtet werden. Ein Beispiel aus dem Trainingscorpus ist:

(2') *We also investigated whether A ... interacts with B ...*

dep-2 Für die ersten vier Regeln werden moderate Generalisierungen zugelassen. Während die Relationen zwischen E1 und CP identisch bleiben, dürfen die zwischen E2 und CP am Anfang zusätzlich eine Abkürzung, Konjunktion und Apposition sowie eine präpositionale Verknüfung oder einen Modifkator einer nominalen Komposition enthalten. Zusätzlich werden Konstruktionen mit

Nr	Type	e1RelToCP/ e1RelToRoot	e2RelToCP	e1RelType e2RelType	CP	e2CP-1	pE1	gpE1
B1	otherCP	nsubj	pobj->prep	ends / equals	IVERB			
B2	e1isCP	pobj->prep	conj	starts / equals			between	INOUN
1	otherCP	nsubj	dobj	equals / eqauls	IVREB			
2	otherCP	nsubj	pobj->prep	equals / eqauls	IVREB	with		
4	otherCP	nsubjpass	pobj->prep	equals / eqauls	IVERB	by		
5	e1isCP	pobj->prep	pobj->prep	starts / equals		by	of	INOUN
8	e1isCP	pobj->prep	conj	starts / equals			between	INOUN
7b	otherCP	pobj->prep	pobj->prep	equals / equals		to	of	INOUN
9	otherCP	nn	pobj->prep	equals / equals	INOUN	with		
Nr	Type	e1RelToCP/ e1RelToRoot	e2RelToCP	e1RelType e2RelType	CP	e2CP-1	isNegation	
12a	e1isCP		nsubj->rcmod	/ ends		IVERB	yes	
12b	e1isCP		rcmod	/ ends		IVERB		
12c	otherCP	nsubjpass	->xcomp	ends / ends		IVERB		
12d	otherCP	nsubj	->xcomp	ends / ends		IVERB		
Nr	Type	e1RelToCP	e1RelType	anywhere		e1RelMax	e2RelMax	
13	e1isCP			IVERBorINOUN			6	
14	otherCP	nsubj	ends	IVERB		3	3	

Abb. 3. Einige der beschriebenen Regeln. Bei e1isCP Regeln wird e1RelToRoot, bei otherCP e1RelToCP verwendet. Nr entspricht der Bezeichnung der Beispiele im Text.

mehreren Verben behandelt, die mit einer Konjunktion am Ende der Relation zwischen E2 und CP abgedeckt werden. Neu gefundene Konstruktionen mit Relationen zwischen A und B sind bspw.:

(1b) *A binds to C and interacts with B.* (2b) *A interacts with another protein, B ...*

dep-3 Je nach Kontext ist die Dependenzausgabe für Konstruktionen wie (5), (6) und (7) nicht immer wie bei S2 in Abb. 2. Stattdessen ist häufig das Interaktionsnomen der gemeinsame Elternknoten. Der Grund hierfür ist, dass der Parser eine PP-Anhängung verschieden durchführen kann. Als Beispiel ist in Abb. 3 Regel 7b angegeben. Zusätzlich werden auch für die Regeln 5, 6 und 7 Generalisierungen zugelassen, während für Regel 8 keine gefunden wurden.

dep-4 Weitere häufig auftretende Konstruktionen werden berücksichtigt:

(9) *A interaction with B* (10) *A is responsible for B expression*

(11) *A-dependent transcription of B*

Bei (9) ist *A interaction*, bei (10) *B expression* jeweils eine nominale Komposition. Die Regel für (9) (Abb. 3) existiert für *with*, *to*, *by* und *of*. Zusätzlich werden auch für diese Regeln Generalisierungen zugelassen.

dep-5 Da der Recall noch immer bei unter 20% liegt, werden weitere Beschränkungen für die Regeln gelockert. Zusätzlich zu den Generalisierungen der Relationen zwischen E2 und CP werden nun auch welche zwischen E1 und CP zugelassen. Dies geschieht teilweise durch Lockerung der strikten Bindung (e1RelType:ends statt equals). Wird diese Relation gelockert, muss die zwischen E2 und CP strikt gelten. Alternativ kann die Relation zwischen E2 und CP gelockert werden, solange die zwischen E1 und CP strikt gilt.

Bei den e1isCP-Regeln, die statt e1RelToCP die Relation zwischen E1 und dem Wurzelknoten betrachten (e1RelToRoot mit e1RelType:start), sind diese Änderungen nicht notwendig. Stattdessen wird erlaubt, dass zwischen E1 und

der Präpositionalphrase (PP) mit dem Interaktionswort eine zusätzliche PP existieren kann. Ein Beispiel, auf das diese Änderung zutrifft, ist:
(6b) *Activation of the subunit of A with B*

Die relevanten Informationen in e1RelToRoot sind nun syntaktisch weiter entfernt von der Entität. Dadurch werden statt der ersten und der zweiten Stelle in der Relation zwischen E1 und dem Wurzelknoten (vgl. 5 in Abb. 3) nun die dritte (ggpE1 statt pE1) und die vierte Stelle (gggpE1 statt gpE1) bedeutsam.

dep-6 Als nächstes werden komplexere syntaktische Formulierungen behandelt, wie Relativsätze, NcI- (Nominativus cum Infinitivo) und Modalverbkonstruktionen. Ist E2 im Relativsatz selbst in subjektivischer Verwendung, sollen die Relativsatzregeln nicht zutreffen. Deshalb werden vor den positiven (vgl. 12b in Abb. 3) negative Regeln (vgl. 12a) aufgerufen, die diese Konstruktionen filtern. Die Regeln für Infinitivkonstruktionen sind 12c und 12d in Abb. 3. Beispiele für diese komplexeren syntaktische Formulierungen sind:
(12a) *... e.g. A, with which B binds C* (12b) *... that A, which activates B ...*
(12c) *A was found to interact with B* (12d) *A was able to interact with B*

Dieses Regelset wird bei der Evaluierung aller Korpora als High-Precision-Regelset (HP-Set) verwendet, um einen Eindruck zu bekommen, wie hoch Recall und Precision bei Verwendung relativ strikter Regeln sind. Das Regelset könnte auf großen Textquellen bereits zu guten Ergebnissen führen, da bei diesen die Precision oft stärker gewichtet wird als der Recall, denn es kann davon ausgegangen werden, dass Relationen mehrmals vorkommen und somit zumindest teilweise mit relativ einfachen Formulierungen.

dep-7 Im letzten Schritt werden sehr allgemeine Regeln hinzugefügt, denn der Recall ist mit 32% noch niedrig und für die Optimierung des F-Scores sollten sich Precision und Recall annähern. Zunächst werden negative Regeln aufgerufen, damit möglichst viele nicht-interagierende Entitätenpaare von der weiteren Betrachtung ausgeschlossen werden. Dann folgen positive Regeln, die auf der Hypothese beruhen, dass Entitäten, die in Bezug auf ihre syntaktischen Relationen eine geringe Distanz aufweisen, miteinander interagieren, sofern zusätzlich ein Interaktionswort innerhalb dieser syntaktischen Relation auftritt. Bei den otherCP Regeln stellt sich heraus, dass für Aktivformulierungen andere Werte verwendet werden sollten als für Passivkonstruktionen. Diese betrachten IVERB und INOUN jeweils einzeln und können je nach Wert von e1RelMax verschiedene Werte für e2RelMax berücksichtigen, während für e1isCP eine sehr allgemeine Regel ausreicht (siehe Regeln 13 und 14 in Abb. 3).

Mit diesen allgemeinen Regeln werden auf dem Trainingscorpus mit dem vollständigen Regelset eine Precision von 48,7%, ein Recall von 50,7% und damit ein F-Score von 49,7% erreicht (siehe Abb. 4). Die Werte auf dem Validierungsset sind ähnlich, der Verlust an Precision von Schritt *dep-6* zu *dep-7* jedoch so groß, dass er nicht vollständig durch den Gewinn an Recall ausgeglichen werden kann.

5 Evaluierung

Für die Evaluierung wurden die von [12] analysierten Korpora (u.a. BioInfer, HPRD50 und LLL05) verwendet. Die Analyse erfolgte mit einem Kookkur-

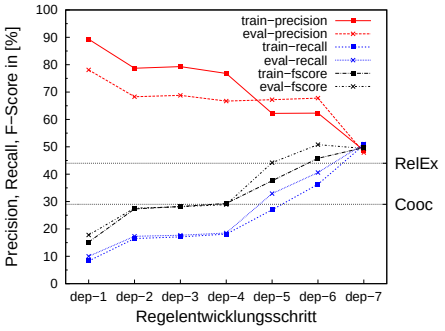

Abb. 4. Stufen der Regelentwicklung für den Dependency Relation Finder auf dem AIMed Korpus: Trainingsset (Quadrate) und Evaluierungsset (Kreuze). Als Vergleich dienen die F-Scores von Cooccurrence und RelEx nach [12].

	BioInfer			HPRD50			LLL05		
	P	R	F	P	R	F	P	R	F
PA	36	56	44	66	60	63	69	48	56
HP	64	25	36	95	38	54	98	41	58
DE	44	39	41	84	56	68	96	72	82
RE	39	45	41	76	64	69	82	72	77
KO	13	99	23	38	100	55	50	100	66

Abb. 5. Evaluierungsergebnisse des UTEMPL Pattern Relation Finders (PA) sowie des Dependency Relation Finders mit High-Precision (HP) und vollständigem Regelset (DE). Als Vergleich dienen die von (Pyysalo et al., 2008) angegebenen Werte für RelEx (RE) und einen Kookkurrenzansatz (KO).

renzansatz und dem bereits erwähnten RelEx, deren Ergebnisse bei der Evaluierung der UTEMPL-Relationsextraktions-Methoden als Vergleichswerte dienen und mit den Ergebnissen in Abb. 5 dargestellt sind[16].

Mit dem Dependency Relation Finder werden F-Score-Werte zwischen 41% und über 82% erreicht, woraus folgt, dass die Korpora sehr verschieden sind. Neben den unterschiedlichen Annotationskriterien für Interaktionen spielen vor allem die Anzahl an Entitäten und Interaktionen pro Satz eine wichtige Rolle für das Abschneiden der Systeme [12]. Die Gegenüberstellung mit den Ergebnissen von RelEx zeigt, das mit UTEMPL wettbewerbsfähige Ergebnisse erzielt werden können. Auffällig ist vor allem, dass mit dem Dep-HP Regelset sehr gute Precision-Werte erzielt werden.

Zusätzlich wurde mit einer Erweiterung des Pattern Relation Finders bei dem BioNLP'09 Shared Task on Event Extraction[17] teilgenommen, und es konnten wettbewerbsfähige Ergebnisse erzielt werden [7].

6 Schlussfolgerung

In diesem Artikel haben wir die hoch flexible, UIMA-basierte Textmining-Pipeline UTEMPL präsentiert, die sich dadurch auszeichnet, dass Software-Komponenten frei kombinierbar eingebunden und Regelsets außerhalb der Software entwickelt werden können. Wir haben zudem aufgezeigt, wie UTEMPL zur Extraktion von Protein-Protein-Interaktionen aus großen Textbeständen biomedizinischer Fachliteratur eingesetzt werden kann. Insbesondere haben wir auf der

[16] Die in den Korpora annotierten Entitäten werden von allen Systemen als gegeben angenommen, damit das Evaluationsergebnis durch Fehler der NER nicht verfälscht wird.

[17] `http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/`

Grundlage entsprechender Evaluationsergebnisse nachgewiesen, dass mit einer tiefen linguistischen dependenz-basierten Methode wettbewerbsfähige Resultate bei der Relationsextraktion erzielt werden können.

Danksagungen. Wir danken Roman Klinger und Theo Mevissen für inhaltliche Diskussionen und den drei anonymen Reviewern für hilfreiche Kommentare.

Literatur

1. Cohen, K.B., Palmer, M., Hunter, L.: Nominalization and Alternations in Biomedical Language. PLoS ONE 3(9):e3158 (2008)
2. Ding, J., Berleant D., Nettleton D., Wurtele E.: Mining MEDLINE: Abstracts, Sentences, or Phrases? In: Proceedings of PSB'02, pp. 326–337 (2002)
3. Friedman, C., Kra, P., Rzhetsky, A.: Two Biomedical Sublanguages: A Description Based on the Theories of Zellig Harris. J Biomed Inform. 35(4), pp. 222–235 (2002)
4. Fundel, K., Küffner, R., Zimmer, R.: RelEx - Relation Extraction Using Dependency Parse Trees. Bioinformatics 23(3), pp. 365–371 (2006)
5. Grishman, R.: Adaptive Information Extraction and Sublanguage Analysis. In: Proceedings of Workshop on Adaptive Text Extraction and Mining at IJCAI-2001, pp. 77–79 (2001)
6. Hanisch, D., Fundel, K., Mevissen, H.T., Zimmer, R., Fluck, J.: ProMiner: Rule-based Protein and Gene Entity Recognition. BMC Bioinformatics, 6 Suppl 1:S14 (2005)
7. Kim, J-D., Ohta, T., Pyysalo, S., Kano, Y., Tsujii, J.: Overview of BioNLP'09 Shared Task on Event Extraction. In: Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task, pp. 1–9 (2009)
8. Lease, M., Charniak, E.: Parsing Biomedical Literature. In: Second International Joint Conference on Natural Language Processing (IJCNLP'05), pp. 58–69 (2005)
9. de Marneffe, M.C., MacCartney, B., Manning, C.D.: Generating Typed Dependency Parses from Phrase Structure Parses. In: Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC), pp. 449–454 (2006)
10. McNaught, J., Black, W.J.: Information Extraction. In: Ananiadou, S., McNaught, J. (eds.) Text Mining for Biology and Biomedicine, ch.7, pp. 143–177. Artech House, Bosten, MA, USA. (2006)
11. Plake, C., Hakenberg, J., Leser, U.: Optimizing Syntax Patterns for Discovering Protein-Protein Interactions. In: Proceedings of the 2005 ACM Symposium on Applied Computing, pp. 195–201 (2005)
12. Pyysalo, S., Airola, A., Heimonen, J., Björne, J., Ginter, F., Salakoski, T.: Comparative Analysis of Five Protein-Protein Interaction Corpora. BMC Bioinformatics, 9 Suppl 3:S6 (2008)
13. Tomanek, K., Wermter, J., Hahn, U.: Sentence and Token Splitting Based on Conditional Random Fields. In: Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING 2007), pp. 49–57 (2007)
14. Wermter, J., Fluck, J., Strötgen, J., Geißler, S., Hahn, U.: Recognizing Noun Phrases in Biomedical Text: An Evaluation of Lab Prototypes and Commercial Chunkers. In: Proceedings of the 1st International Symposium on Semantic Mining in Biomedicine, pp. 25–33 (2005)
15. Zhou, D., He, Y.: Extracting Interactions between Proteins from the Literature. J Biomed Inform. 41(2), pp. 393–407 (2008)