Institut für Informatik der
Friedrich-Schiller-Universität Jena

Automatische Extraktion von Protein-Protein-Interaktionen in biomedizinischen Texten unter Verwendung von Supportvektormaschinen

Diplomarbeit
zur Erlangung des akademischen Grades
Diplom-Informatiker

vorgelegt von
Erik Fäßler

betreut von
Prof. Dr. Birgitta König-Ries und
Prof. Dr. Udo Hahn

27. März 2009

Kurzfassung

Diese Arbeit beschäftigt sich mit der automatischen Extraktion von Protein-Protein-Interaktionen (PPI) in biomedizinischen Texten. Dafür wird ein Verfahren des maschinellen Lernens eingesetzt, die sogenannte Supportvektormaschine (SVM). Eine SVM klassifiziert Objekte im Merkmalraum durch eine klassentrennende Hyperebene. Dieses Verfahren kann durch sogenannte Kernoperatoren erweitert werden, die die zu klassifizierenden Objekte in einen höherdimensionalen Raum abbilden, wo sie besser linear voneinander getrennt werden können. Es werden verschiedene Kernoperatoren zur Lösung des Problems der PPI-Extraktion betrachtet. Das beste Verfahren ist der Graph-Kern, der 55,33% *F*-Maß auf dem AIMed-Corpus erreicht und 62,99% *F*-Maß auf dem Genereg-Corpus.

Abstract

This thesis' topic is given by the automated extraction of protein-protein-interaction (PPI) in biomedical literature. A technique of the field of machine learning is employed, the so-called support vector machine (SVM). An SVM classifies objects in feature space by means of a seperating hyperplane. An extension to this approach is given by the kernel-trick. Kernel functions are used to map object vectors into higher dimensional space, where a seperating hyperplane is found more easily. Several kernel functions for the task of PPI-extraction are considered. The best results are obtained by the graph kernel. It reaches 55,33% *F*-Score with the AIMed-Corpus and 62,99% *F*-Score with the Genereg-Corpus.

Inhaltsverzeichnis

KAPITEL 1

Einleitung

Die größte Datenbank von Texten der biomedizinischen Domäne stellt MEDLINE[1] dar. Seit 2005 werden der Sammlung täglich 2.000 bis 4.000 neue Dokumente hinzugefügt, im Herbst 2008 verfügte MEDLINE über 16.888.640 Dokumente (PUBMED 2008). Davon waren 2008 alleine 671.904 neu hinzugekommen. Wegen der großen Datenmengen sind zentrale, mächtige Suchmechanismen nötig, die den Zugriff auf die Dokumente ermöglichen. Für die gezielte Suche ist es weiterhin von Vorteil, die in den Dokumenten beschriebenen Informationen so darzustellen, dass sie automatisch abgefragt werden können. So wäre es beispielsweise wünschenswert, direkt nach bestimmten Proteinen, Interaktionen zwischen Proteinen oder Ereignissen suchen zu können, in die die gesuchten Stoffe involviert sind. Eben wegen der bestehenden Größendimensionen ist es nicht möglich, eine solche Annotation der Daten manuell durchzuführen. Es werden deshalb maschinelle Ansätze benötigt, um die Informationen von Interesse zu extrahieren.

Der Fokus dieser Arbeit liegt auf Relationen zwischen Proteinen und Genen, spezieller auf der automatischen Extraktion von so genannten Protein-Protein-Interaktionen (PPI). Zur Lösung dieser Aufgabe werden einerseits *regelbasierte Ansätze* verwendet (Fundel u. a. 2006). Bei solchen Methoden werden von Experten wie Biomedizinern und Linguisten Sätze von Regeln aufgestellt. Anhand dieser Regeln sollen die gesuchten Informationen identifiziert und ggfs. klassifiziert werden. Eine zweite Möglichkeit bieten die Ansätze des *maschinellen Lernens*. Die verfügbaren Methoden stellen eine Fülle an Algorithmen dar, die mit unterschiedlichen Grundprinzipien automatisch Regelsätze lernen. Diese maschinell gewonnenen Regeln entsprechen dabei zumeist keinen Regeln, wie sie ein Mensch formuliert hätte. Statt dessen werden mathematische Modelle zur Erkennung und Klassifikation der gesuchten Elemente der Fachliteratur berechnet. Für das automatische Lernen werden Daten benötigt, die Beispiele der Objekte von Interesse

[1] Auf die Sammlung kann über PUBMED von `http://www.ncbi.nlm.nih.gov/pubmed/` aus zugegriffen werden.

enthalten, für die ihre Klassenzugehörigkeit bekannt ist. Solche *Corpora* werden zudem verwendet, um ein System der Informationsextraktion zu evaluieren. In dieser Arbeit werden *Supportvektormaschinen* verwendet. Diese Technik ist auf Grund der hohen Generalisierbarkeit des Lernalgorithmus' und der Gestaltungsfreiheit von *Kernoperatoren* eine bevorzugte Wahl im Feld der PPI-Extraktion.

Der Vergleich zweier Systeme zur Protein-Protein-Interaktion (PPI)-Extraktion gestaltet sich dabei als schwierig. Unterschiedliche Prozessierungsweisen der Daten und Auswertungsstrategien der Methoden verwehren den Blick darauf, wie sich die Systeme untereinander verhalten. Deshalb werden die konkreten, hier eingesetzten Methoden in einem *zentralen Projekt reimplementiert* und die Auswertung findet stets auf die gleiche Art und Weise statt. Hierdurch können aussagekräftige Schlüsse über die Leistungen der einzelnen Ansätze gezogen werden.

In Kapitel 2 wird die Aufgabenstellung der PPI-Extraktion detaillierter erläutert. Kapitel 3 führt in die Grundbegriffe der Musteranalyse sowie der Klassifikation ein und beschreibt das automatische Lernverfahren, das für diese Arbeit gewählt wurde. Die verschiedenen Methoden zur Merkmalextraktion und die verwendeten Kernoperatoren, die zusammen mit dem Lernverfahren zur PPI-Extraktion verwendet werden, sind in Kapitel 4 beschrieben. In Kapitel 5 wird erläutert, wie die Systemauswertung geschieht, welche Probleme damit bezüglich des Systemvergleichs verbunden sind und auf welche Weise freie Parameter selektiert werden. Das Datenmaterial, das zum Anlernen und Evaluieren einer konkreten Methode dient, wird in Kapitel 6 vorgestellt. Kapitel 7 geht auf Details der Implementierung der Methoden ein, die in Kapitel 8 ausgewertet und miteinander verglichen werden.

KAPITEL 2

Aufgabenstellung

Informationsextraktion (IE) bezeichnet die Aufgabe, unstrukturierte semantische Informationen aus Texten in strukturierte Daten zu verwandeln (Jurafsky und Martin 2008). Die resultierenden Daten gehören dabei typischerweise einer vordefinierten Klasse bzw. Kategorie an. Dadurch gliedert sich die Verarbeitung im Allgemeinen in die Erkennung von relevanten Informationen und deren anschließende Klassifikation in die korrekte Kategorie. Gegeben sei folgendes Textbeispiel aus einem Internetmagazin (*Wirtschafts-Woche* 2008):

> ' Der Mobilfunkkonzern *Vodafone* will noch im Frühjahr den Vorstand seiner Festnetztochter *Arcor* umbauen und dabei Arcor-Chef *Harald Stöber* schrittweise entmachten. '

Um die Aussage dieses Textes zu ermitteln, sind mehrere Schritte nötig. Zunächst müssen die beteiligten Parteien, etwa Firmen und Personen, identifiziert werden. In diesem Fall geht es um die Firma *Vodafone*, eine zweite Firma *Arcor* und eine Person mit dem Namen *Harald Stöber*. Im Anschluss sind die Beziehungen zwischen diesen Parteien und schließlich das beschriebene Ereignis oder die Ereignisse selbst zu extrahieren. Es sollte festgestellt werden, dass *Arcor* in einer '**gehört-zu**'-Relation mit *Vodafone* steht und dass *Harald Stöber* durch die '**Chef-von**'-Relation mit *Arcor* verbunden wird. Der Text teilt außerdem die Ereignisse des Umbaus von *Arcor* und den rückläufigen Machtbesitz von *Harald Stöber* mit. Damit ergeben sich als Unteraufgaben der Informationsextraktion die Erkennung von benannten Entitäten (engl. Named Entity (NE)[1]), hier also etwa Personen und Firmen, Relationen zwischen den Entitäten, sowie die Erkennung von Ereignissen und zeitlichen Ausdrücken.

In dieser Arbeit liegt der Fokus auf der Erkennung und Klassifikation binärer Relationen, also Beziehungen zwischen zwei Entitäten. Um eine Relation zu finden,

[1] Der englische Begriff wird im deutschen Sprachraum akzeptiert und oft verwendet, weshalb in dieser Arbeit auch mit der englischen Originalversion gearbeitet wird.

müssen zunächst die potentiellen Argumente identifiziert werden. Deshalb geht der Relationsextraktion ein Erkennungsprozess der im Text enthaltenen Entitäten voraus.

Bei der **Relationsextraktion** werden semantische Relationen zwischen den zuvor gefundenen Entitäten in einem Text ermittelt und klassifiziert. Wie oben bereits angedeutet, geht es dabei meist um binäre Relationen, die in eine kleine zuvor festgelegte Menge von Kategorien eingeteilt werden sollen. Typische allgemeine Relationen, wie sie etwa im Zeitungscorpus ACE (Doddington u. a. 2004) annotiert werden, sind u.a.

- Familienbeziehungen
- Angestelltenbeziehungen
- Teil-Ganzes-Beziehungen

In der Biomedizin werden häufig Beziehungen zwischen Genen und Proteinen gesucht, in diesem Zusammenhang wird von **Protein-Protein-Interaktion (PPI)** gesprochen. Die Extraktion von PPI stellt damit die Übertragung der allgemeinen Relationsextraktion auf den biologischen bzw. biomedizinischen Bereich dar. Im Allgemeinen sollen Interaktionen zwischen Genen und Proteinen identifiziert werden. Es existiert allerdings keine allgemein anerkannte oder befolgte Richtlinie, wie PPIs zu annotieren sind Pyysalo u. a. (2008). Die u.a. daraus resultierende Problematik der Ergebnisvergleichbarkeit wird näher in Kapitel 5.1 besprochen. Im Folgenden werden typische Erscheinungsformen von PPI vorgestellt. Sie lassen sich prinzipiell weiter in Untertypen einteilen, beispielsweise

- Aktivierung (einer Genexpression)
- Inhibition (einer Genexpression)
- Bindung (eines Proteins an DNA)
- Regulation (einer Genexpression)
- Transkription (von DNA in RNA)

Es kommen auch weitere Relationstypen zum Einsatz, wie etwa die oben im allgemeineren Kontext genannte Teil-Ganzes-Beziehung. Der LLL05-Challenge-Corpus (Nédellec 2005) definiert die Zugehörigkeit eines Gens zu einer bestimmten Regulon-Familie ebenfalls als eine zu extrahierende Relation. Fundel u. a. (2006) formalisieren diese Beziehungen:

- Effektor-Relation-Ziel (A aktiviert B)

- Relation-von-Ziel-durch-Effektor (Aktivierung von A durch B)
- Relation-zwischen-Effektor-und-Ziel (Interaktion zwischen A und B)[2]

Ein konkreter Satz aus der Biomedizin mit einer einfachen Relation hat etwa folgende Gestalt:

> ' Recombinant *hTAFII250* binds directly to *TBP* both in vitro and in yeast, and participates in the formation of the TFIID complex. '

Es wird die Bindung des Proteins *hTAFII250* an *TBP* beschrieben. In einem Satz können auch mehrere Interaktionen beschrieben werden. Nicht selten trifft man dabei auf *Koordinationen*:

> ' A 51-residue region from the conserved C-terminal region of *TBP*, previously shown to be the binding site for the viral activator protein *E1A*, interacts with *c-Fos* and *c-Jun* proteins. '

Hier wird eine Interaktion von *TBP* mit den Proteinen *E1A*, *c-Fos* und *c-Jun* beschrieben, wobei *TBP* nur ein einziges mal erwähnt wird. *TBP* ist eine Abkürzung für '*TATA box-binding protein*'. Das heißt, eine einzelne Entität kann aus mehreren Wörtern bzw. Tokens bestehen. Diese Tatsache ist für das maschinelle Auffinden von Relationen - und zuvor auch für die Named Entity Recognition (NER) - ein wichtiger Punkt, der die Textprozessierung erschweren kann. Im Folgenden Satz ist ein Beispiel für eine NE zu finden, die aus mehreren Tokens besteht:

> ' The ability of *c-Fos* and *c-Jun* proteins to interact directly with the *TATA box-binding protein* (*TBP*), the general transcription factor required for initiating the assembly of transcription complexes, was investigated. '

Zudem gibt es auch hier eine Koordination, allerdings in anderer "Richtung" als im vorigen Beispiel. Standen oben die Proteinbezeichnungen, die die Koordination enthalten, am Ende des Satzes, so stehen sie hier am Anfang und beziehen sich auf das erst noch zu nennende *TBP*. Dieses Beispiel zeigt, dass die Anzahl der Relationen in einem Satz oder Text von der Interpretation des Datenmaterials abhängt. Der obige Satz enthält nach der Annotation im AIMed-Corpus (Bunescu u. a. 2005) insgesamt vier Relationen: *c-Fos* und *c-Jun* interagieren jeweils sowohl mit *TATA box-binding protein* als auch mit der Abkürzung *TBP*. Zudem bleibt der Aufgabenstellung überlassen, ob Relationen asymmetrisch aufgefasst und erkannt werden müssen oder nicht. Beispielsweise kommt die Identifikation von Passivausdrücken der Aufgabe gleich, nicht nur die Interaktion sondern auch Effektor und Ziel derselben zu bestimmen. So ist es etwa in der Aufgabenstellung des LLL05 Challenge festgelegt, in AIMed jedoch nicht.

[2] Übersetzung aus dem Englischen durch den Verfasser dieser Arbeit.

Das Ziel dieser Arbeit ist die *Erstellung einer Bibliothek* von Implementierungen verschiedener Strategien für die *Extraktion von PPIs* auf Basis von *Supportvektor-Klassifikation*. Die zu klassifizierenden Objekte sind dabei durch *Paare von Entitäten* gegeben, die als potentielle *Argumente* einer PPI aufgefasst werden. Typischerweise beschränkt man sich dabei auf Entitätenpaare, deren Elemente beide aus dem gleichen Satz stammen. Das bedeutet, dass für einen Satz mit n Entitäten $\binom{n}{2}$ Objekte - Entitenpaare - gebildet werden, die Gegenstand für die Klassifikation sind. Zudem sollen die verschiedenen Methoden ausgewertet und miteinander verglichen werden. Die hier verwendeten Algorithmen sind in der Lage, verschiedene Ansätze miteinander zu kombinieren. Durch die Identifikation von Stärken und Schwächen der unterschiedlichen Lösungsmöglichkeiten wird nach Kombinationen gesucht, die die Ergebnisse der PPI-Extraktion verbessern.

KAPITEL 3

Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen umrissen, die die Basis für die Relationsextraktion bilden, wie sie im Rahmen dieser Arbeit durchgeführt wird. Allgemein ist es zunächst nötig, gegebene natürlichsprachige Texte aufzubereiten, so dass durch den Computer gezielt auf Informationen zugegriffen werden kann. In Abschnitt 3.1 werden hierfür übliche Methoden skizziert. Für die Erkennung von interessanten Informationen im so aufbereiteten Text werden Methoden der Musteranalyse und -erkennung verwendet. Die formalen Grundlagen dafür werden in Abschnitt 3.2 geliefert. In Abschnitt 3.3 schließlich wird die Klassifikationstechnik vorgestellt, von der in dieser Arbeit Gebrauch gemacht wird.

3.1 Textstrukturierung

Ein maschinenlesbarer Text besteht in seiner grundlegendsten Computerdarstellung aus einer Folge einzelner Zeichen. Es liegen zunächst keine Informationen darüber vor, was ein Wort oder ein Satz ist. Ebenso ist unbekannt, ob Entitäten im Text enthalten sind und um welche es sich ggfs. handelt. Bevor allerdings mit der Erkennung und Klassifikation von Relationen begonnen werden kann, müssen mindestens diese Informationen vorliegen. Um noch tiefere und detailliertere Informationen über den vorliegenden Text zu gewinnen, wird oft von Strukturanalysen wie Chunking und Parsing Gebrauch gemacht, die im Folgenden näher erklärt werden. Doch auch - oder gerade - diese Informationen müssen zunächst generiert bzw. extrahiert werden. Für eine Aufgabe wie die syntaktische Strukturanalyse eines Satzes muss zuerst bekannt sein, was ein Satz ist und die Erkennung von NEs setzt eine Tokenisierung voraus. Daher wird häufig ein Ansatz für die Textverarbeitung gewählt, bei der die Aufgaben von “unten nach oben” durchgeführt werden.

Eine typische Reihenfolge wäre etwa (Hahn und Wermter 2006):

1. Satzerkennung
2. Tokenisierung
3. Wortartenerkennung (Part-of-Speech (PoS)-Tagging)
4. Erkennung von Entitäten (Named Entity Recognition)
5. Chunking
6. Parsing

Die durch diese Verarbeitungsschritte gewonnenen Informationen können anschließend für semantische Prozessierungen wie die Relationsextraktion verwendet werden. Im Folgenden werden die einzelnen Schritte kurz erläutert.

Die ***Satzerkennung*** ist dafür zuständig, eine Textsegmentierung auf Satzebene durchzuführen. Die Aufgabe besteht im Wesentlichen aus der Desambiguierung von Punkten in einem Text. Es muss ermittelt werden, ob ein gegebener Punkt einen Satz-Ende-Punkt oder etwa einen Abkürzungspunkt (Dr. Schmidt), einen Dezimaltrenner (42.471) oder ähnliches darstellt. Das größte Problem stellen die Abkürzungspunkte dar, da sie sich am Wortende befinden und deshalb schwieriger zu identifizieren sind als etwa ein Punkt in einer Dezimalzahl. Es reichen allerdings bereits einfache Filteralgorithmen und reguläre Ausdrücke, um auf einem allgemeinsprachlichen Corpus wie dem Brown-Corpus eine Satzerkennungsrate von 99,7% zu erreichen (Grefenstette und Tapanainen 1994).

Tokenisierung meint die Segmentierung des Textes auf Wortebene und geschieht typischerweise auf Satzebene. Was dabei als Token aufzufassen ist, bleibt in Detailfragen dem Anwendungsfall überlassen. Im Allgemeinen wird mit einem Token allerdings etwas Wortähnliches gemeint sein. Abweichende Definitionen gibt es für den Umgang mit Zahlenwerten, Bindestrichkonstrukten und Ähnlichem. Gerade im biomedizinischen Bereich kann es von Bedeutung sein zu entscheiden, wie etwa mit Bindestrichen umzugehen ist, die in den Namen von Proteinen und Genen vorkommen[1].

Die ***Wortartenerkennung*** bzw. das *Part-of-Speech (PoS)-Tagging* baut direkt auf der Tokenisierung auf. An jedes Token wird eine Marke (Tag und Label) vergeben, die die

[1] Man betrachte etwa die Zeichenkette *IL-2-induced*. Es wäre möglich, an allen Bindestrichen Tokengrenzen zu ziehen, wodurch aber der Zugriff auf die zusammengehörige Gruppe *[IL-2]* erschwert würde, die ein Gen bzw. Protein bezeichnet. Die Entscheidung, ob ein Bindestrich eine Tokengrenze darstellt oder nicht, wird durch solche Phänomene erschwert.

genaue Wortart bestimmt. Die Menge von Marken - das Tagset - ist dabei nicht auf die üblichen linguistischen Wortarten begrenzt, sondern enthält oft auch Tags für einzelne Zeichen wie Bindestriche, Schrägstriche, Klammern und weitere Zeichen, die je nach Anwendungsfall als eigene Tokens gesehen werden.

Als Erwähnung einer ***Named Entity*** werden im Allgemeinen alle Wortvorkommen definiert, die etwas bezeichnen, das durch einen Eigennamen referenziert werden kann. Die Erkennung solcher Entitäten in einem Text wird als ***Named Entity Recognition*** bezeichnet. Aus praktischen Gründen werden häufig auch weitere Textelemente wie Gewichts- oder Datumsangaben als NE-Erwähnungen[2] gezählt. An dieser Stelle wird klar, dass die Definition von NEs vom praktischen Zweck der jeweiligen Anwendungen abhängig ist. Eine allgemein anerkannte Definition für Zeitungstexte liefern die Aufgabenstellungen der Message Understanding Conferences (MUC) (*MUC-7 Named Entity Task Definition* 1997):

- Organisationen, Personen, Orte (Entitätennamen)
- Datums- und Zeitangaben (Zeitausdrücke)
- Geldwerte, Prozentangaben (Zahlenausdrücke)

Zusätzlich wird oft ermittelt, ob und welche NEs das gleiche Objekt in der realen Welt referenzieren. Die Zusammefassung solcher als "gleich" definierte Named Entitys wird als Referenzauflösung bezeichet.

Andere Arten von Entitäten findet man etwa im biomedizinischen Bereich. Die interessanten Entitäten sind hier Protein- oder Gennamen wie "Interleukin-2" oder "IL2". Dabei sind die Bezeichnungen für ein Gen und das zugehörige Protein oft äußerst ähnlich, was die zuverlässige Einteilung einer Entität in die richtige Kategorie erschwert. Das eben genannte Beispiel etwa wurde der UnitProt-Datenbank (*UniProt* 2009) entnommen. Die Kurzversionen der Namen von Gen und Protein unterscheiden sich im gegebenen Fall lediglich in der Verwendung des Bindestrichs.

Chunking bezeichnet die Zusammenfassung von zusammengehörigen Wortgruppen zu Phrasen. Es handelt sich dabei nicht notwendigerweise um eine vollständige Textsegmentierung wie bei der Satzerkennung oder Tokenisierung. Es werden wichtige Phrasen wie Nominal-, Verbal- oder Präpositionalphrasen identifiziert, wobei nicht jedes Wort in einem Satz einer Phrase zugeordnet werden muss und statt dessen als "nicht-zugehörig" markiert werden kann.

[2] Obwohl genaugenommen zwischen einer Entität und ihrer Erwähnung in einem Text zu unterscheiden ist, wird der Einfachheit halber beides im Folgenden als "Named Entity" oder "Entität" bezeichnet.

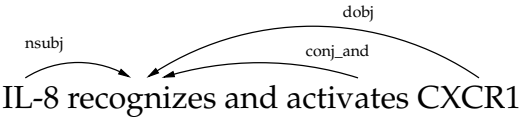

Abbildung 3.1: Ein Beispielsatz, der gemäß des Stanford-Schemas analysiert wurde. Die Dependenzkanten tragen als Beschriftung die syntaktische Beziehung vom Dependenten zum Kopf der Relation.

Die tiefgreifendste syntaktische Analyse eines Satzes ist das ***Parsing***. Es setzt Wörter und Phrasen miteinander in Beziehung und führt damit eine syntaktische Desambiguierung durch. Es gibt zwei Arten des Parsings, die für die Relationsextraktion bislang von Bedeutung sind: zum Einen die Analyse basierend auf einer *Phrasenstrukturgrammatik (PSG)* und zum Anderen eine Analyse, die eine Dependenzgrammatik zur Basis hat. Eine PSG kennt neben den Wörtern selbst - die sogenannten Terminale - eine geschlossene Menge von abstrakten Elementen, die Nichtterminale. Die Nichtterminale erfüllen im Allgemeinen die Funktion von Phrasen, fassen also Wörter zu Gruppen zusammen. Dabei kann eine Phrase wiederum andere Phrasen enthalten, so dass es in einem Ableitungsbaum längere Pfade von abstrakten Kategorien geben kann. Die Nicht-Wort-Elemente einer *Dependenzgrammatik* gehören einer geschlossenen Menge von binären Relationen an, die jeweils zwei Wörter direkt miteinander in Beziehung setzen. Die Relationen sind dabei im Allgemeinen nicht symmetrisch, so dass ein gerichteter Graph entsteht. In den meisten Dependenzformalismen hat dieser Ableitungsgraph eine Baumstruktur. Es gibt allerdings ebenfalls Repräsentationen, die die Kreisfreiheit zerstören (Marneffe u. a. 2006) oder sogar mehrere Zusammenhangskomponenten bilden, indem nicht alle Wörter durch eine Relation mit dem Rest des Satzes verbunden werden (Sleator und Temperley 1993).

Ein einfaches Beispiel einer Dependenzanalyse eines Satzes aus der Biomedizin ist in Abbildung 3.1 gegeben. Es ist deutlich zu erkennen, dass das Prädikat *recognizes* ein zentraler Bezugspunkt der Gesamtstruktur ist. Die Wörter des Satzes werden durch die Dependenzkanten miteinander in Beziehung gesetzt, so dass die Verbindung zweier Wörter direkt eine syntaktische Bedeutungszuweisung erhält. So zeigt etwa die Kante *IL-8* $\xrightarrow{nsub}$ *recognizes* unmittelbar an, dass *IL-8* das Subjekt (nominales Subjekt, nsubj) bezüglich des Satzprädikats darstellt. Die Kante *CXCR1* $\xrightarrow{dobj}$ *recognizes* verdeutlicht die Objektrolle (direktes Objekt, dobj) von *CXCR1*. Es gibt dabei keine allgemeingültige Konvention für die Kantenrichtung. Das heißt, es ist nicht festgelegt, ob der Dependent auf den Dependenzkopf zeigt oder anders herum. Durch die Strukturbildung nahe an der grammatischen Satzstruktur werden syntaktische Informationen

unmittelbar verfügbar, die die Satzstruktur sehr direkt beschreiben und im Bereich der PPI-Extraktion erfolgreich eingesetzt werden. Das Beispiel aus Abbildung 3.1 wurde gemäß des Stanford-Schemas geparst. Das bedeutet, dass Struktur und Kantenbeschriftungen bestimmten Regeln folgen, die in Marneffe u. a. (2006) beschrieben sind. Dependenzparser unterscheiden sich vor allem in Anzahl und Art der Dependenzrelationen. Während die Link-Grammatik, die für die syntaktische Analyse des LLL05-Corpus eingesetzt wurde (Sleator und Temperley 1993), nur 5 Relationstypen kennt, definiert das Stanford-Schema eine Hierarchie von 48 Relationstypen. Hierarchisch bedeutet hier, dass etwa eine Subjekt-Kante in einen spezielleren Fall der Subjekt-Relation wie nominales Subjekt oder passives, nominales Subjekt differenziert werden kann.

3.2 Musteranalyse

(nach Schukat-Talamazzini 2005)

In diesem Abschnitt werden die formalen Grundlagen gegeben, die letztlich zur Klassifikation führen. Der allgemeine Fall ist die Musteranalyse. Sie bezeichnet die symbolische Beschreibung von Objekten bzw. Mustern. Diese Beschreibung repräsentiert optimal das vorhandene Wissen und muss dabei optimal zum vorhandenen Signal passen (Niemann 1990). Diese Formulierung wird im Folgenden mathematisch gefasst.

Definition 3.1. *Die Umwelt ist die Gesamtheit aller Größen, die sich mit physikalischen Geräten messen lassen. Damit ergibt sich die Umwelt als Menge*

$$\mathfrak{U} = \{ {}^{\rho}\boldsymbol{b}(\boldsymbol{x}) \mid \rho = 1, 2, \dots \}$$

von Funktionen $\boldsymbol{b}(\boldsymbol{x})$.[3]

Bei einer gegebenen Anwendung wird man sich im Allgemeinen auf einen Ausschnitt der Umwelt beschränken. Diese Fokussierung ergibt den Problemkreis $\boldsymbol{\Omega}$, der nur Objekte bzw. Funktionen innerhalb des Fokus' enthält.

Definition 3.2. *Ein Problemkreis* $\boldsymbol{\Omega}$ *enthält nur Objekte bzw. Funktionen, die zu einem konkreten, strikt begrenzten Anwendungsfall gehören. Er ist definiert durch die Menge*

$$\boldsymbol{\Omega} = \{ {}^{\rho}\boldsymbol{f}(\boldsymbol{x}) \mid \rho = 1, 2, \dots \} \subset \mathfrak{U}$$

von Funktionen ${}^{\rho}\boldsymbol{f}(\boldsymbol{x})$ *und ist eine Teilmenge der Umwelt* $\mathfrak{U}$.

[3] Zur besseren und eindeutigeren Lesbarkeit werden Vektorgrößen im Folgenden fett geschrieben.

Damit ergibt sich unmittelbar die Definition eines Objektes im Sinne der Musteranalyse:

Definition 3.3. *Die Elemente des Problemkreises* $\mathbf{\Omega}$ *heißen Muster. Ein Muster ist folglich eine Funktion*

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{cases} f_1(x_1, x_2, \ldots, x_m) \\ f_2(x_1, x_2, \ldots, x_m) \\ \ldots \\ f_n(x_1, x_2, \ldots, x_m) \end{cases}$$

Die Funktionen $f_1, f_2, \ldots, f_m$ *beschreiben die Attribute des Musters und heißen Merkmal bzw. Feature*

Ein Objekt ist damit ein Vektor von Funktionswerten. Es setzt sich im abstraktesten Sinne aus n Kanälen zusammen, die von jeweils m Variablen abhängig sind. Eine Videosequenz besitzt beispielsweise je einen Kanal für die Grundfarben, etwa Rot, Grün und Blau. Der Farbwert eines Bildpunkts im Videostrom hängt dabei von seiner Position (x, y) im Bild ab, sowie vom Zeitpunkt t, dem das Bild zugeordnet wurde. Im Feld des Natural Language Processing (NLP) sind die den Mustern zugrunde liegenden Objekte Textelemente, wie etwa Tokens oder Strukturelemente wie Dependenzrelationen. Um eine Darstellung als Zahlenwertevektor zu erhalten, werden typischerweise Indikatorfunktionen eingesetzt, die dem Element eine bestimmte Eigenschaft zu- bzw. absprechen. Bei der Tokenisierung ist etwa von Interesse, ob ein Textelement groß geschrieben ist oder ob es ausschließlich aus Buchstaben (im Gegensatz zu Zahlen) besteht. Es sind aber auch Merkmalfunktionen mit größerem Wertebereich wie etwa eine Angabe der Elementlänge denkbar. Je nach Anwendung werden auch ganze Wörter oder längere Zeichenketten als Merkmal verwendet. Das bedeutet, dass für jedes Wort eine Indikatorfunktion existiert, die den Wert Eins annimmt, falls das Wort beobachtet wurde, und sonst Null zurück gibt. Ist eine Gewichtung eines Wortes gewünscht, können auch andere Werte verwendet werden. Soweit nicht anders beschrieben, wird in dieser Arbeit davon ausgegangen, dass in solchen Fällen der Wert Eins vergeben wird. Konkrete Merkmalfunktionen in Verbindung mit der PPI sind in Kapitel 4 gegeben.

Unter *Klassifikation* wird nun eine Zuordnungsvorschrift verstanden, die ein Objekt $\boldsymbol{x} \in \boldsymbol{\Omega}$ einer von K *Klassen* zuweist. Manchmal wird auch von $K+1$ Klassen ausgegangen. Die zusätzliche Kategorie dient dann als Rückweisungsklasse, wenn der Klassifikator etwa keine sichere Entscheidung treffen kann. Sie wird aber auch als Sammelbecken für Objekte verwendet, die sich zwar im Problemkreis befinden, aber kein Objekt von Interesse darstellen. Für das System von Klassen folgt dann:

Definition 3.4. *Sei $\boldsymbol{\Omega}$ ein Problemkreis, $K \in \mathbb{N}$ und das Mengensystem $(\Omega_1, \ldots, \Omega_K)$ eine Partition über $\boldsymbol{\Omega}$, d.h. es gilt:*

$$\begin{aligned} \Omega_\kappa &\neq \emptyset, \qquad \kappa = 1, \ldots, K \\ \Omega_\kappa \cap \Omega_\lambda &= \emptyset, \qquad \kappa \neq \lambda \\ \bigcup_{\kappa=1}^{K} \Omega_\kappa = \boldsymbol{\Omega} \quad &\textit{bzw.} \quad \bigcup_{\kappa=1}^{K+1} \Omega_\kappa = \boldsymbol{\Omega} \end{aligned}$$

Dann heißen die $\Omega_\kappa, \kappa = 1, \ldots, K$ Klassen bzw. Klassengebiete von $\boldsymbol{\Omega}$.

Der Problemkreis $\boldsymbol{\Omega}$ enthält bei Klassifikationsanwendungen mehrere, voneinander getrennte Klassen. Es ist zwar denkbar, dass ein Objekt im Problemkreis zu mehreren Klassen zugleich gehört, es wird in dieser Arbeit aber von disjunkten Klassengebieten ausgegangen.

Auf Grund dieser Definitionen wird im Folgenden die Klassifikationsmethode beschrieben, die Verwendung in dieser Arbeit findet.

3.3 Supportvektormaschinen

Durch Definition 3.3, die die mathematische Beschreibung eines Musters liefert, wird implizit ein Merkmal- oder Featureraum definiert. Dieser Raum besitzt für jede Attributsfunktion der Muster aus dem Problemkreis $\boldsymbol{\Omega}$ eine Dimension, ist also m-dimensional. Die Werteskala jeder Achse ergibt sich durch die Skala ihrer zugehörigen Funktion. Wird von reellen Funktionswerten der Attributsfunktionen ausgegangen, nimmt der Merkmalsraum die Form eines Vektorraums über dem Körper der reellen Zahlen an.

Eine Supportvektormaschine ist ein statistisches Verfahren zur Klassifikation von unbekannten Objekten auf Basis von Beispieldaten. In ihrer Grundform löst sie das Zwei-Klassen-Problem. Zu diesem Zweck berechnet der Lernalgorithmus der Supportvektormaschine (SVM) eine *Hyperebene* $\mathcal{H}$ im Merkmalsraum, die die Instanzen des Lernmaterials voneinander trennt. Ein neues Objekt wird anschließend auf Grund seiner Position relativ zur Hyperebene klassifiziert. Da die Hyperebene den Merkmals-

raum in zwei Teile partioniert, gibt es genau zwei Klassen. In Unterabschnitt 3.3.1 wird die prinzipielle Funktionsweise eines solchen Lern- und Klassifikationsalgorithmus' anhand eines einfachen Beispiels deutlich gemacht. Unterabschnitt 3.3.2 erklärt auf dieser Grundlage die genaue Funktionsweise im Falle einer SVM. In Unterabschnitt 3.3.3 schließlich wird eine Technik beschrieben, mit der eine SVM besser an ihre Lerndaten angepasst werden kann und auf der die Experimente beruhen, die in dieser Arbeit durchgeführt wurden.

3.3.1 Einführung

(nach Schölkopf und Smola 2001)

In diesem Abschnitt wird ein einfacher Lernalgorithmus zur Klassifikation von Mustern aus zwei Klassen beschrieben. Er soll die grundsätzliche Funktion von Klassifikation mittels einer Hyperebene im Merkmalsraum verdeutlichen, bevor die Details für den Fall einer SVM erläutert werden.

Es sei $\mathbf{\Omega} \subset \mathbb{R}^n$ und $\omega = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m) \} \subset \mathbf{\Omega} \times \{ -1, +1 \}$ eine klassifizierte *Stichprobe* aus $\mathbf{\Omega}$. Das heißt, zu jedem Muster $\boldsymbol{x}$ aus ω ist seine Klasse -1 oder $+1$ bekannt. Es soll ein Algorithmus beschrieben werden, der mittels der Stichprobe ω eine Hyperebene $\mathcal{H}$ konstruiert. Diese Trennfläche unterliege der Bedingung, an jedem Punkt gleichweit von beiden *Klassenzentren* der in der Stichprobe enthaltenen Klassen entfernt zu sein. Die Klassenzentren werden als Mittelwertvektoren der Stichprobenelemente beschrieben, die der jeweiligen Klasse angehören. Um neue, bisher unbekannte Objekte einer der beiden Kategorien -1 oder $+1$ zuzuweisen, wird für einen entsprechenden Eingabevektor seine Lage relativ zur Hyperebene $\mathcal{H}$ berechnet, die die Klassenzentren voneinander trennt.

Zunächst müssen dafür die Mittelwertvektoren $\boldsymbol{c}_+$ und $\boldsymbol{c}_-$ der Klassen $+1$ und -1 berechnet werden:

$$\boldsymbol{c}_+ = \frac{1}{m_+} \sum_{\{ i | y_i = +1 \}} \boldsymbol{x}_i, \tag{3.3.1}$$

$$\boldsymbol{c}_- = \frac{1}{m_-} \sum_{\{ i | y_i = -1 \}} \boldsymbol{x}_i \tag{3.3.2}$$

Dabei seien m_+ und m_- die Anzahl der positiven bzw. negativen Beispiele. Keine der Klassen sei leer, so dass $m_+, m_- > 0$ gelte. Ein unbekanntes Objekt $\boldsymbol{x} \in \mathbb{R}^n$ wird nach seiner Nähe zu den Mittelwertvektoren klassifiziert und der Klasse y zugewiesen, deren Klassenzentrum näher an $\boldsymbol{x}$ liegt. Dieses Prinzip wird durch Abbildung 3.2 dargestellt. Es werde zunächst der Vektor $\boldsymbol{w} \stackrel{def}{=} \boldsymbol{c}_+ - \boldsymbol{c}_-$ betrachtet, der die Klassenzentren miteinander verbindet. Der Punkt $\boldsymbol{c} \stackrel{def}{=} (\boldsymbol{c}_+ + \boldsymbol{c}_-)/2$ liegt auf $\boldsymbol{w}$, genau in der Mitte

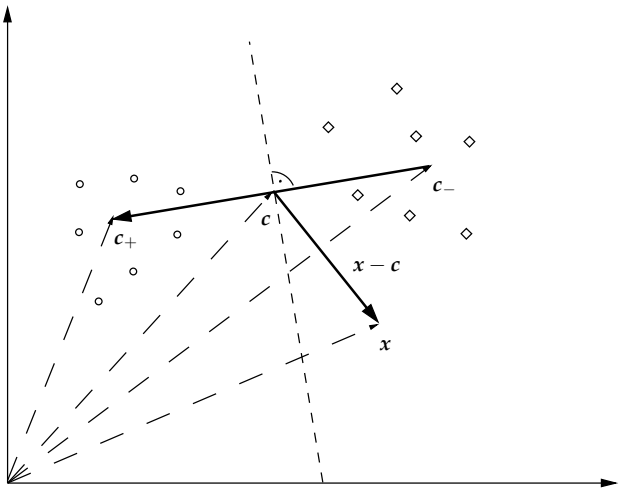

Abbildung 3.2: Ein einfacher Algorithmus zur Berechnung einer Hyperebene, die die Klassen „Kreis“ und „Karo“ voneinander trennt. Der Normalenvektor $\boldsymbol{w}$ ergibt sich durch die Verbindung der Klassenzentren $\boldsymbol{c}_+$ und $\boldsymbol{c}_-$ der Stichprobe ω. Um zu entscheiden, welcher Klasse ein zu klassifizierender Vektor $\boldsymbol{x}$ angehört, wird der Winkel zwischen $\boldsymbol{w}$ und dem Verbindungsvektor vom Mittelpunkt zwischen den Klassenzentren zu $\boldsymbol{x}$ selbst betrachtet. Ist der Winkel kleiner $\pi/2$, so wird $\boldsymbol{x}$ als Karo klassifiziert, sonst als Kreis. Die auf diese Weise induzierte Hyperebene ist als gestrichelte Linie dargestellt.

zwischen den Mittelwerten $\boldsymbol{c}_+$ und $\boldsymbol{c}_-$. Der zu klassifizierende Vektor $\boldsymbol{x}$ wird durch $\boldsymbol{x} - \boldsymbol{c}$ mit $\boldsymbol{c}$ und damit auch $\boldsymbol{w}$ verbunden. Um zu bestimmen, welcher der Mittelwertvektoren $\boldsymbol{x}$ am nächsten ist, muss nun berechnet werden, ob der Winkel zwischen den Vektoren $\boldsymbol{x} - \boldsymbol{c}$ und $\boldsymbol{w}$ kleiner als $\pi/2$ ist. Es ergibt sich

$$\begin{aligned} y &= \operatorname{sgn}\langle(\boldsymbol{x} - \boldsymbol{c}), \boldsymbol{w}\rangle \\ &= \operatorname{sgn}\langle(\boldsymbol{x} - (\boldsymbol{c}_+ + \boldsymbol{c}_-)/2), \boldsymbol{w}\rangle \qquad (3.3.3) \\ &= \operatorname{sgn}(\langle \boldsymbol{x}, \boldsymbol{c}_+\rangle - \langle \boldsymbol{x}, \boldsymbol{c}_-\rangle + b) \qquad (3.3.4) \end{aligned}$$

mit der Verschiebung

$$b \stackrel{def}{=} \frac{1}{2}(\|\boldsymbol{c}_-\|^2 - \|\boldsymbol{c}_+\|^2).$$

Dabei bezeichne $\langle\cdot,\cdot\rangle$ das Skalarprodukt zweier Vektoren und es sei die Vektornorm $\|\cdot\|$ durch $\|\boldsymbol{x}\| \stackrel{def}{=} \sqrt{\langle \boldsymbol{x}, \boldsymbol{x}\rangle}$ gegeben. Die Gleichung 3.3.4 ergibt sich durch die elementweise Umformung des Skalarproduktes aus Gleichung 3.3.3, wodurch man auch die Verschiebung b erhält. Gleichung 3.3.4 induziert eine Entscheidungsgrenze in der Gestalt einer Hyperebene $\mathcal{H}$, wie in Abbildung 3.2 illustriert wird.

Das Ziel ist jedoch eine *Entscheidungsfunktion*, die direkt durch die Lernvektoren der Stichprobe ω definiert ist und für ein gegebenes Objekt $\boldsymbol{x} \in \mathbb{R}^n$ dessen Klasse -1

oder $+1$ zurück gibt. Dazu werden die Definitionen aus den Gleichungen 3.3.1 und 3.3.2 in die Gleichung 3.3.4 der Entscheidungsgrenze eingesetzt. Man erhält die Entscheidungsfunktion

$$y = \operatorname{sgn}\left(\frac{1}{m_+}\sum_{\{\,i|y_i=+1\,\}}\langle \boldsymbol{x}, \boldsymbol{x}_i\rangle - \frac{1}{m_-}\sum_{\{\,i|y_i=-1\,\}}\langle \boldsymbol{x}, \boldsymbol{x}_i\rangle + b\right) \tag{3.3.5}$$

und die Verschiebung b wird auf gleiche Weise zu

$$b \overset{def}{=} \frac{1}{2}\left(\frac{1}{m_-^2}\sum_{\{\,(i,j)|y_i=y_j=-1\,\}}\langle \boldsymbol{x}, \boldsymbol{x}_i\rangle - \frac{1}{m_+^2}\sum_{\{\,(i,j)|y_i=y_j=+1\,\}}\langle \boldsymbol{x}, \boldsymbol{x}_i\rangle\right).$$

Damit ist ein Klassifikator formuliert, der neue Objekte durch eine Partition des Merkmalsraums klassifiziert. Diese Partition ist durch eine Hyperebene $\mathcal{H}$ mit dem *Normalenvektor* $\boldsymbol{w}$ und der *Verschiebung* b gegeben.

Durch Gleichung 3.3.5 wird aber lediglich der Zwei-Klassen-Fall gelöst. Im allgemeinen Fall von K Klassen wird von K Entscheidungsfunktionen ausgegangen, die für ein gegebenes Objekt $\boldsymbol{x}$ bestimmen, ob es zur Klasse $\kappa \in \{\,1,\ldots,K\,\}$ gehört oder nicht. Ein Vektor von Entscheidungsfunktionen gibt dann für ein gegebenes Muster seine Klassenzugehörigkeit an:

Definition 3.5. *Sei $\boldsymbol{\Omega} \subset \mathbb{R}^n$ ein Problemkreis mit K Musterklassen. Eine Abbildung $\boldsymbol{\delta}$*

$$\boldsymbol{\delta} : \begin{cases} \mathbb{R}^n & \to \mathbb{R}^{K+1} \\ \boldsymbol{x} & \mapsto \boldsymbol{\delta}(\boldsymbol{x}) = (\delta_1(\boldsymbol{x}), \delta_2(\boldsymbol{x}), \ldots, \delta_{K+1}(\boldsymbol{x}))^\top \end{cases}$$

heißt Entscheidungsregel, falls die Normalisierungsbedingung

$$\sum_{\lambda=1}^{K+1}\delta_\lambda(\boldsymbol{x}) = 1, \quad (\kappa = 1,\ldots,K)$$

gilt. Die Regel heißt scharf, wenn alle $\boldsymbol{\delta}(\boldsymbol{x})$ Einheitsvektoren sind. Andernfalls heißt die Regel unscharf oder randomisiert.

Generell ist es möglich, eine Wahrscheinlichkeit für die Zugehörigkeit für jede Klasse anzugeben. Diese Wahrscheinlichkeitsangabe kann für so genannte Schwellwertverfahren verwendet werden. Dabei wird die Klassenzugehörigkeit nach Berechnung aller Entscheidungsfunktionswerte einer Nachprozessierung unterzogen, die beispielsweise auf Grund einer sehr knappen Entscheidung die Klassenzugehörigkeit eines Objekts in eine unterlegene Klasse wechselt. Dies kann je nach Verteilung der Daten zu besseren Ergebnissen eines Klassifikators führen.

Im Falle des hier beschriebenen Klassifikators und auch im Falle von SVMs müssten solche Wahrscheinlichkeitsangaben allerdings zunächst hergeleitet und gesondert berechnet werden. Ein natürlicheres Maß für die Entscheidungssicherheit im Zusammenhang mit einer SVM ist die Entfernung eines Vektors $\boldsymbol{x}$ zur klassentrennenden Hyperebene. In Abschnitt 5.3 ist näher beschrieben, wie in dieser Arbeit mit Schwellwerten gearbeitet wird.

3.3.2 Beschreibung der SVM

Im vorigen Abschnitt wurde ein einfacher Algorithmus beschrieben, der durch eine Hyperebene im Merkmalsraum Klassifikationsentscheidungen für bislang unklassifizierte Objekte $\boldsymbol{x} \in \boldsymbol{\Omega}$ trifft. Dieser Algorithmus ist der prinzipiellen Funktionsweise einer SVM bereits sehr ähnlich. Der wesentliche Unterschied liegt in den Anforderungen, die an die Hyperbene $\mathcal{H}$ gestellt werden. Die Ebene wurde zuvor lediglich durch ihre Position relativ zu den Klassenzentren in der Stichprobe $\omega \subset \boldsymbol{\Omega}$ beschrieben. Die Hyperebene orientiert sich in ihrer Lage im Raum nicht direkt an den Lernvektoren sondern nur an den Klassenzentren, was eine fehlende Verallgemeinerungsfähigkeit des Klassifikators zur Folge hat. So kann es vorkommen, dass sogar eine linear separable Lernstichprobe durch den obigen Algorithmus nicht vollständig voneinander getrennt wird, also bereits bei der Reklassifikation der Stichprobe Fehler auftreten (vgl. Abbildung 3.3). Eine solche Hyperebene ist aus zwei Gründen nicht optimal: zum Einen wegen des unnötigen Reklassifikationsfehlers und zum Anderen, weil ein *maximaler Abstand* zu allen Lernvektoren anstrebenswert ist, solange die Klassen voneinander getrennt bleiben. Diese Eigenschaft ist für die in Abbildung 3.3 skizzierte Ebene offenbar nicht gegeben. Durch die Maximierung dieses *Randes* - von Hyperebene zu nächsten Lernvektoren - wird die hohe Generalisierung[4] der SVM-Klassifikation erreicht, da beide Klassen einen maximal großen Toleranzbereich für Ausreißer besitzen (vgl. Abbildung 3.4).

Eine mathematische Formulierung dieser Forderungen an $\mathcal{H}$ ist durch

$$\underset{\boldsymbol{w}\in\mathbb{R}^n, b\in\mathbb{R}}{\operatorname{argmax}} \min \{ \|\boldsymbol{x} - \boldsymbol{x}_i\| \mid \boldsymbol{x} \in \mathbb{R}^n, \langle w, x\rangle + b = 0, i = 1, \ldots, n \}$$

mit dem Normalenvektor $\boldsymbol{w}$ und Verschiebung b der Hyperebene $\mathcal{H}$ gegeben. Die Bedingung $\langle w, x\rangle + b = 0$ für $\boldsymbol{x} \in \mathbb{R}$ beschreibt dabei alle Punkte $\boldsymbol{x}$, die genau auf $\mathcal{H}$ liegen.

[4] Der Vorgang, aus Beispieldaten möglichst allgemeingültige Regeln zu lernen, die auch für unbekannte Daten zutreffen, wird als Generalisierung bezeichnet.

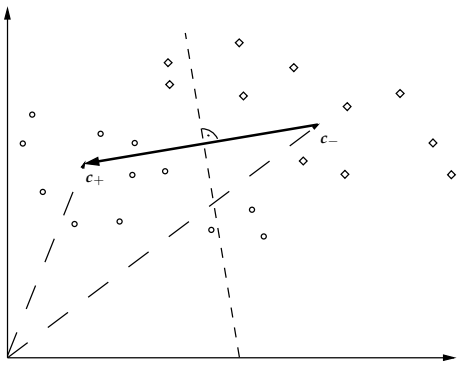

Abbildung 3.3: Eine nichtoptimale Hyperebene zur Trennung von zwei Klassen. Die Ausrichtung der Ebene orientiert sich nur an den Klassenzentren und lässt die Klassenvektoren selbst außen vor.

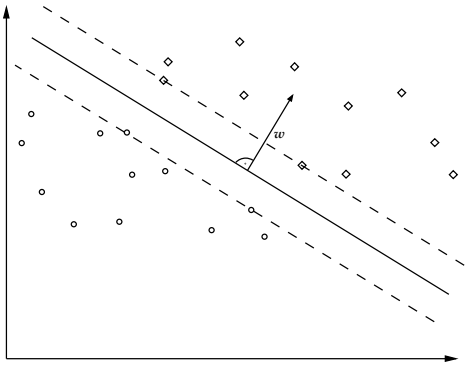

Abbildung 3.4: Eine optimale separierende Hyperebene. Gestrichelt eingezeichnet ist der Rand der Hyperebene zu den Vektoren, die der Ebene am nächsten liegen. Es gibt keinen Reklassifikationsfehler des Lernmaterials.

Dieses Ergebnis erhält man durch Umformung der Differenz von Skalarprodukten aus Gleichung 3.3.4:

$$\begin{aligned} y &= \operatorname{sgn}(\langle \boldsymbol{x}, \boldsymbol{c}_+ \rangle - \langle \boldsymbol{x}, \boldsymbol{c}_- \rangle + b) \\ &= \operatorname{sgn}(\langle \boldsymbol{x}, \boldsymbol{w} \rangle + b) \end{aligned} \tag{3.3.6}$$

Steht $\boldsymbol{x}$ orthogonal auf der Normalen $\boldsymbol{w}$, so verschwindet das Skalarprodukt $\langle \boldsymbol{x}, \boldsymbol{w} \rangle$. Verliefe $\mathcal{H}$ durch den Ursprung, würden genau diese zu $\boldsymbol{w}$ orthogonalen Vektoren auf der Ebene liegen. Liegt $\mathcal{H}$ nicht auf dem Ursprung, so muss nach Berechnung des Skalarprodukts die Verschiebung addiert werden, so dass alle $\boldsymbol{x}$ mit $\langle \boldsymbol{x}, \boldsymbol{w} \rangle + b = 0$ auf der Ebene $\mathcal{H}$ liegen.

Im Folgenden wird beschrieben, wie Normalenvektor und Verschiebung der optimalen Hyperebene $\mathcal{H}$ einer SVM berechnet werden. Es sei analog zu Unterabschnitt 3.3.1 $\omega = \{ (\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m) \} \subset \boldsymbol{\Omega} \times \{ -1, +1 \}$ eine klassifizierte Stichprobe aus dem Problemkreis $\boldsymbol{\Omega}$. Um die optimale, klassenseparierende Hyperebene $\mathcal{H}$ zu finden, muss folgendes quadratisches Programm gelöst werden:

$$\underset{\boldsymbol{w} \in \mathbb{R}^n, b \in \mathbb{R}}{\text{minimiere}}\ \tau(\boldsymbol{w}) = \frac{1}{2} \|\boldsymbol{w}\|^2 \tag{3.3.7}$$

$$\text{bezüglich } y_i(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b) \geq 1 \text{ für alle } i = 1, \ldots, m \tag{3.3.8}$$

Die *Längenminimierung des Normalenvektors* sichert zusammen mit den Nebenbedingungen $y_i(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b) \geq 1$ den *maximal großen Rand* von der Ebene zu den nächsten

Lernvektoren. Dies beruht auf der Tatsache, dass $y_i(\langle \boldsymbol{x}_i, \boldsymbol{w}\rangle + b)/\|\boldsymbol{w}\|$ gleich der Entfernung des Punktes $\boldsymbol{x}_i$ zur Hyperebene ist. Dieses Ergebnis erhält man durch den Zusammenhang $\langle \boldsymbol{x}, \boldsymbol{w}\rangle = x'\|\boldsymbol{w}\|$, wobei x' die Länge der Projektion von $\boldsymbol{x}$ auf $\boldsymbol{w}$ bezeichne. Diese Länge ist gerade der Abstand von $\boldsymbol{x}$ zu $\mathcal{H}$, wenn eine orthogonale Verbindung von $\boldsymbol{x}$ zur Ebene gezogen wird. Falls $b \neq 0$ muss noch die Verschiebung der Hyperebene berücksichtig werden. Da die Lernvektoren, die der Ebene $\mathcal{H}$ am nächsten liegen, die Bedingung $|(\langle \boldsymbol{x}_i, \boldsymbol{w}\rangle + b)| = 1$ erfüllen (siehe Gleichung 3.3.8), folgt ein Ebenenabstand ihrerseits von $1/\|\boldsymbol{w}\|$, d.h. der Rand hat eine Größe von $1/\|\boldsymbol{w}\|$. Das Minimieren des Normalenvektors resultiert daher im größtmöglichen Rand.

Um das Optimierungsproblem 3.3.7 mit den Nebenbedingungen 3.3.8 zu lösen, werden die Lagrangemultiplikatoren $\alpha_i \geq 0$ eingeführt:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i(\langle \boldsymbol{x}_i, \boldsymbol{w}\rangle + b) - 1)$$

Das Funktional L soll bezüglich der so genannten primären Variablen $\boldsymbol{w}$ und $\boldsymbol{b}$ minimiert und bezüglich den α_i, die als duale Variablen bezeichnet werden, maximiert werden. Dies entspricht dem Auffinden eines Sattelpunktes des Funktionals. [5] An diesem Sattelpunkt verschwinden die partiellen Ableitungen bezüglich der primären Variablen. Es ergibt sich

$$\frac{\partial}{\partial b} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = 0 \quad \text{und} \quad \frac{\partial}{\partial \boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = 0,$$

woraus

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{3.3.9}$$

sowie

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \tag{3.3.10}$$

folgen.

Die Darstellung 3.3.10 des Normalenvektors für die optimale Hyperebene entspricht einer Linearkombination von Vektoren aus einer Teilmenge der Lernstichprobe.

[5] Im Folgenden werden einige Details und Hintergründe der Optimierungsfrage selbst nur kurz erwähnt oder weggelassen. An dieser Stelle geht es um das Finden einer optimalen, trennenden Hyperebene. Die Details der Optimierung können in Schölkopf und Smola (2001) nachgelesen werden.

Es wird im Allgemeinen nicht das gesamte Lernmaterial verwendet, was für die Festlegung von Ausrichtung und Position der Hyperebene nicht notwendig ist. Das liegt daran, dass ein Teil der Lagrangemultiplikatoren α_i verschwindet, wenn die entsprechenden Lernvektoren $\boldsymbol{x}_i$ für die Beschreibung von $\mathcal{H}$ nicht benötigt werden.

Es sind genau die Lagrangemultiplikatoren ungleich Null, deren Vektoren der resultierenden Hyperebene am nächsten sind. Um das zu sehen, betrachte man die Bedingungen 3.3.8. Ist eine solche Bedingung verletzt, d.h. $y_i(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b) - 1 < 0$ für ein $i \in \{1, \ldots, n\}$, könnte das gesamte Funktional beliebig groß werden, indem der zugehörige Lagrangemultiplikator α_i inkrementiert wird. Das heißt, α_i ist ungleich Null. Durch die Zielfunktion 3.3.7 werden aber der Normalenvektor $\boldsymbol{w}$ sowie die Verschiebung b so angepasst, dass die i-te Bedingung wieder gilt, d.h. $y_i(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b) - 1 = 0$. Lernvektoren, deren Lagrangemultiplikator einen Wert ungleich Null besitzt, heißen *Supportvektoren* und legen die Lage sowie die Position der Hyperebene fest.

Für jede Bedingung, für die $y_i(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b) - 1 > 0$ gilt, muss α_i hingegen gleich Null sein, um eine Maximierung von L zu erreichen. Dadurch fallen diejenigen Vektoren aus der Linearkombination 3.3.10 weg, die sich im „inneren“ des Klassengebietes befinden und dementsprechend weiter von der Trennebene entfernt sind als andere Lernvektoren mit gleicher Klassenzugehörigkeit.

Um nun Normalenvektor und Verschiebung der Ziel-Hyperebene in der Praxis zu berechnen, wird eine Darstellung benötigt, die nicht von eben diesen Größen abhängt. Das wird durch die Formulierung des so genannten *dualen Optimierungsproblems* erreicht, das man durch Substitution der Gleichungen 3.3.9 sowie 3.3.10 in das Funktional L gewinnt:

$$\begin{aligned} &\underset{\boldsymbol{\alpha} \in \mathbb{R}^n}{\text{maximiere}}\, W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\ &\text{bezüglich } \alpha_i \geq 0 \text{ für alle } i = 1, \ldots, n \text{ und } \sum_{i=1}^{m} \alpha_i y_i = 0. \end{aligned} \tag{3.3.11}$$

Durch Lösung dieses quadratischen Programms erhält man eine Hyperebene $\mathcal{H}$ mit den gewünschten, oben beschriebenen Eigenschaften.

Bis zu dieser Stelle wurde das prinzipielle Vorgehen für die Berechnung einer optimalen separierenden Hyperebene zur Lösung des Zwei-Klassen-Problems einer SVM beschrieben. Die obigen Verfahren gelten jedoch nur für den Fall von *linear separablen* Daten. Für den Fall von Nichtseparierbarkeit werden Schlupfvariablen $\xi_i \geq 0$ eingeführt, die Falschklassifikation während der Trainingsphase bestrafen, um eine Ebene zu finden, die neben der Länge des Normalenvektors die Strafsummen minimiert. Dazu werden die ξ_i zum Gegenstand der Minimierung in der Zielfunktion 3.3.7. Man

erhält

$$\tau(\boldsymbol{w}, \boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{m}\xi_i, \tag{3.3.12}$$

wobei $C > 0$ eine Konstante darstellt, mit der zwischen der Maximierung des Ebenenrandes und der Minimierung des Trainingfehlers abgewogen werden kann. In praktischen Anwendungen ist selten eine gute Wahl für C a priori bekannt und wird beispielsweise durch Experimente ermittelt. Das weitere Vorgehen für die Berechnung einer optimalen Hyperebene unter Berücksichtigung von Schlupfvariablen gestaltet sich analog zu obigen Ausführungen und soll hier nicht gesondert betrachtet werden.

3.3.3 Kernel-Trick

In Unterabschnitt 3.3.2 wurde die Berechnung einer optimalen Trennebene $\mathcal{H}$ im Merkmalsraum $\mathbb{R}^n$ der Lernstichprobe $\omega = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$ beschrieben. Für den Fall der linearen Nichtseparierbarkeit der Lerndaten wurden Strafterme eingeführt, die wie die Parameter von $\mathcal{H}$ Gegenstand der Optimierung waren. Dennoch wird lediglich eine lineare Trennfläche im Merkmalsraum beschrieben. Selbst wenn die nicht linear separierbaren Daten durch eine einfache nichtlineare Funktion voneinander getrennt werden könnten, etwa durch ein Polynom, so wäre man bei aktuellem Stand der Verfahrensbeschreibung nicht in der Lage, die Daten fehlerlos voneinander zu trennen. In diesem Abschnitt wird deshalb der so genannte *Kernel-Trick* vorgestellt. Dazu wird eine Klasse von Funktionen - so genannten *Kernoperatoren* oder kurz *Kerne* bzw. *Kernels* - beschrieben, die in der Lage sind, implizit *Skalarpodukte* von Vektoren in extrem *hochdimensionalen* Vektorräumen zu berechnen. Dabei werden Vektoren aus dem Merkmalsraum implizit in den höherdimensionalen Raum nichtlinear abgebildet. Dadurch wird es möglich, eine trennende Hyperebene im hochdimensionalen Raum zu berechnen, in dem die Lerndaten potentiell besser voneinander trennbar sind.

Zunächst soll die einfachere Trennbarkeit in einem Raum höherer Dimension intuitiv verdeutlicht werden. Abbildung 3.5 (links) zeigt eine Datensituation, in der die Klassen nicht linear voneinander trennbar sind. Durch Abbildung der Daten in einen Raum mit höherer Dimension - hier wird beispielhaft die Dimension z_3 hinzugefügt - kann es erreicht werden, dass die Klassen „hintereinander“ liegen, also auf einer der neuen Raumachsen klar voneinander abgrenzbar sind. Diese Situation ist in Abbildung 3.5 (rechts) skizziert. Sie zeigt die Ausgangssituation „von der Seite“. Durch Rückabbildung der Vektoren in den ursprünglichen Raum entsteht eine nichtlineare Trennfunktion im Merkmalsraum (Abbildung 3.6). Es folgt eine mathematische Be-

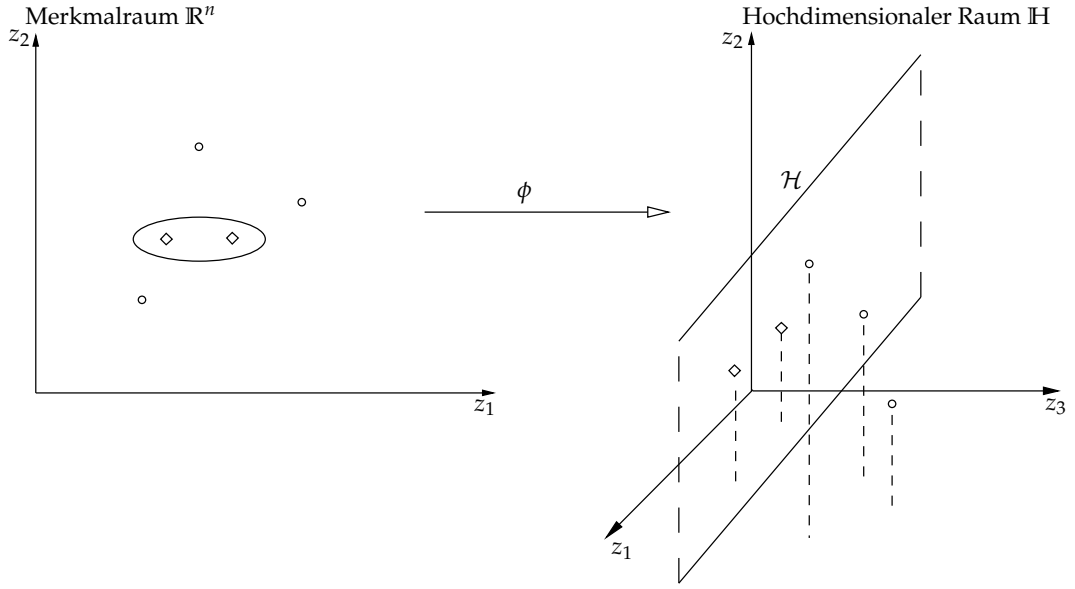

Abbildung 3.5: Nicht linear separierbare Datensituation im Merkmalsraum $\mathbb{R}^n$ (links). Durch nichtlineare Einbettung in den hochdimensionalen Raum $\mathbb{H}$ mittels der Abbildung ϕ kann eine separierende Hyperebene in $\mathbb{H}$ gefunden werden.

schreibung dieser Vorgehensweise, um auch nichtlineare Trennfunktionen im Merkmalsraum zu erreichen.

Man betrachte eine nichtlineare Funktion ϕ, die einen Vektor $\boldsymbol{x} \in \mathbb{R}^n$ in einen hochdimensionalen Raum $\mathbb{H}$ abbildet:

$$\phi : \begin{cases} \mathbb{R}^n & \to \mathbb{H} \\ \boldsymbol{x} & \mapsto \phi\boldsymbol{x} \end{cases}$$

Ein Kernoperator K ist dann eine symmetrische Funktion

$$K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R},$$

wobei $K(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi\boldsymbol{x}, \phi\boldsymbol{y} \rangle$ mit $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}$ gelten soll. K berechnet damit das Skalarprodukt der nichtlinearen Einbettung von $\boldsymbol{x}$ und $\boldsymbol{y}$ in den Raum $\mathbb{H}$.

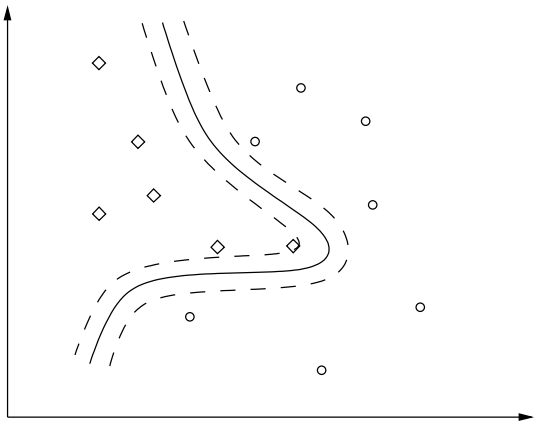

Abbildung 3.6: Nichtlineare Trennfunktion im Merkmalsraum $\mathbb{R}^n$. Durch Rückabbildung der linear separierten Vektoren $\phi\boldsymbol{x}$ in den Merkmalsraum entsteht eine nichtlineare Klassentrennung. Der Verlauf der Trennfunktion ist als durchgezogene Linie dargestellt, der Rand zu den nächsten Lernvektoren als gestrichelte Linien. Die Datensituation ist die gleiche wie in Abbildung 3.5, es wurden lediglich einige Datenpunkte zur Verdeutlichung hinzugefügt.

Damit die Gleichheit zwischen Kernoperator und hochdimensionalem Skalarprodukt tatsächlich gegeben ist, muss K weiterhin die so genannte *Mercer-Bedingung* erfüllen, die in folgendem Satz beschrieben wird:

Satz 3.1. *(Mercer-Bedingung) Erfüllt ein symmetrischer Operator* $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ *für jede Funktion* $g : \mathbb{R}^n \to \mathbb{R}$ *mit* $\int g^2(\boldsymbol{x})d\boldsymbol{x} < \infty$ *die Bedingung*

$$\int\int K(\boldsymbol{x},\boldsymbol{y})g(\boldsymbol{x})g(\boldsymbol{y})d\boldsymbol{x}d\boldsymbol{y} \geq 0,$$

so gibt es einen (un)endlichdimensionalen Vektorraum $\mathbb{H}$ *mit dem Skalarprodukt* $\langle\cdot,\cdot\rangle$ *und eine Einbettung* $\mathbb{R}^n \to \mathbb{H}$ *mit*

$$K(\boldsymbol{x},\boldsymbol{y}) = \langle\phi\boldsymbol{x},\phi\boldsymbol{y}\rangle = \sum_i (\phi\boldsymbol{x})_i(\phi\boldsymbol{y})_i, \quad \forall\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^n$$

.

Erfüllt ein beliebiger Operator K nicht die Mercer-Bedingung, so entsprechen die Funktionwerte $K(\boldsymbol{x},\boldsymbol{y})$ nicht dem Skalarprodukt $\langle\phi\boldsymbol{x},\phi\boldsymbol{y}\rangle$ der expandierten Vektoren. Das würde den folgenden Verfahren die mathematische Grundlage und damit ihre Gültigkeit entziehen.

Die Mercer-Bedingung für eine gegebene Funktion $K(\cdot,\cdot)$ zu zeigen, ist allgemein schwierig. Es liegen jedoch einige Ergebnisse vor, die in der Praxis regulär Anwendung finden.

So erfüllt jeder *Polynomialkern*

$$K(\boldsymbol{x},\boldsymbol{y}) = (\gamma \cdot \langle \boldsymbol{x},\boldsymbol{y}\rangle + r)^d \text{ mit } \gamma, r \in \mathbb{R}$$

die Mercer-Bedingung. Der triviale lineare Kernoperator $\langle \boldsymbol{x},\boldsymbol{y}\rangle$ ist offenbar ein Spezialfall des Polynomialkerns. Ebenfalls oft in praktischen Anwendungen verwendet ist der *Gaußkern* (Radialbasisfunktion *(RBF)*):

$$K(\boldsymbol{x},\boldsymbol{y}) = e^{-\|x-y\|^2/2\gamma^2} \text{ mit } \gamma \in \mathbb{R}, \tag{3.3.13}$$

der mit allen Parametern γ die Mercer-Bedingung erfüllt.

Zudem ist die Klasse der (gültigen) Kernoperatoren bezüglich linearer Kombination abgeschlossen: Erfüllen alle Kernoperatoren $K_i : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, $i \in \mathbb{N}$, die Mercer-Bedingung, so auch

$$K(\boldsymbol{x},\boldsymbol{y}) \overset{def}{=} \sum_{i=0}^{\infty} c_i \cdot K_i(\boldsymbol{x},\boldsymbol{y}), \quad c_i > 0(\forall i), \tag{3.3.14}$$

falls $K(\cdot,\cdot)$ uniform konvergent ist. Durch dieses Ergebnis lassen sich verschiedene Kerne miteinander kombinieren, um neue, gültige Kerne zu bilden. Diese Technik wird, soweit möglich und sinnvoll, im Rahmen dieser Arbeit verwendet, um bessere Ergebnisse zu erhalten.

Eine weitere Abschlusseigenschaft von Kernoperatoren betrifft deren *Normalisierung*. Erfüllt der Kernoperator $K(\cdot,\cdot)$ die Mercer-Bedingung, so auch der normalisierte Kernoperator $\tilde{K}(\boldsymbol{x},\boldsymbol{y})$, dessen Werte durch die Transformation

$$\tilde{K}(\boldsymbol{x},\boldsymbol{y}) = \frac{K(\boldsymbol{x},\boldsymbol{y})}{\sqrt{K(\boldsymbol{x},\boldsymbol{x})K(\boldsymbol{y},\boldsymbol{y})}} \tag{3.3.15}$$

gewonnen werden. Es gilt $\tilde{K}(\boldsymbol{x},\boldsymbol{x}) = 1$ bzw. allgemein $0 \leq \tilde{K}(\boldsymbol{x},\boldsymbol{y}) \leq 1$. Die Normalisierung entspricht der Abbildung aller Vektoren auf eine Einheitssphäre, wodurch die Daten gleichmäßig im Raum verteilt werden. Es bleibt bislang allerdings die Frage offen, wie diese Ergebnisse und Techniken bei der Berechnung einer nichtlinearen Trennfunktion im Merkmalsraum bzw. einer separierenden Hyperebene im Raum $\mathbb{H}$ behilflich sein können und soll im Folgenden beantwortet werden. Die Entscheidungsfunktion 3.3.6 wurde für die Klassifikation von neuen Objekten durch die SVM verwendet. Sie lautet wie folgt:

$$f(\boldsymbol{x}) = \operatorname{sgn}(\langle \boldsymbol{x},\boldsymbol{w}\rangle + b).$$

Die SVM-Hyperebene hat durch die Formulierung des Lagrange-Funktionals die Gestalt $\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i$ erhalten (vgl. Gleichung 3.3.10). Durch Einsetzen dieser Beschreibung der Hyperebene in die Entscheidungsfunktion 3.3.6 kann man diese wie folgt

umformen:

$$\begin{aligned} f(\boldsymbol{x}) &= \operatorname{sgn}\left(\langle \boldsymbol{x}, \boldsymbol{w}\rangle + b\right) \\ &= \operatorname{sgn}\left(\langle \boldsymbol{x}, \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i\rangle + b\right) \\ &= \operatorname{sgn}\left(\sum_{i=0}^{m} \alpha_i y_i \langle \boldsymbol{x}, \boldsymbol{x}_i\rangle + b\right). \end{aligned}$$

die Lernvektoren und das unbekannte Objekt gehen ausschließlich durch das Skalarprodukt in die Entscheidungsfunktion ein. Durch einen gültigen Kernoperator - d.h. er erfüllt die Mercer-Bedingung - fällt daher die explizite, und im Allgemeinen unbekannte, Abbildung in $\mathbb{H}$ der beteiligten Vektoren weg. Auf diese Weise ist es möglich, die Daten implizit in einem höherdimensionalen Raum $\mathbb{H}$ zu separieren, was sich als nichtlineare Trennfunktion im Merkmalsraum manifestiert. Diese Technik ist allgemein einsetzbar und nicht auf SVMs beschränkt. Jede in einem Vektorraum operierende Methode kann davon Gebrauch machen, wenn sie sich auf die Berechnung von Skalarprodukten beschränken kann.

Für die praktische Anwendung des Kernel-Tricks für die Klassifikation ist von großem Belang, dass es nicht notwendig ist, explizit Skalarprodukte zwischen Vektoren zu berechnen. Für die Berechnung der Hyperebene $\mathcal{H}$ reicht die Angabe der bloßen Skalare - also Zahlen aus $\mathbb{R}$ - aus. Diese Tatsache hat weitreichende Konsequenzen für die allgemeine Methodik. So ist es nicht mehr zwingend nötig, überhaupt eine vektorielle Repräsentation der Objekte zu liefern, mit denen Training - das Lernen der Hyperebene - oder Klassifikation durchgeführt werden sollen. Das Skalarprodukt zweier Vektoren kann als *Ähnlichkeitsmaß* ausgedrückt durch den euklidischen Abstand in einem Vektorraum aufgefasst werden. Es ist ebenso möglich, ein beliebiges anderes Ähnlichkeitsmaß $K(\cdot,\cdot)$ zu definieren, das auf einer anderen Repräsentation als Vektoren operieren kann. Die Bedingungen bei dieser Methode, um gültige Kernwerte zu erhalten, sind die Symmetrie und die positive Semidefinitheit der Operation, die sich über die Matrix definiert, welche im Folgenden eingeführt wird. Dadurch resultieren als mindestens nötige Eingabe für die SVM so genannte *Kern-Matrizen* oder auch *Gram-Matrizen* der Form

$$\begin{pmatrix} K(\boldsymbol{x}_1,\boldsymbol{x}_1) & K(\boldsymbol{x}_1,\boldsymbol{x}_2) & \dots & K(\boldsymbol{x}_1,\boldsymbol{x}_m) \\ K(\boldsymbol{x}_2,\boldsymbol{x}_1) & K(\boldsymbol{x}_2,\boldsymbol{x}_2) & \dots & K(\boldsymbol{x}_2,\boldsymbol{x}_m) \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ K(\boldsymbol{x}_m,\boldsymbol{x}_1) & K(\boldsymbol{x}_m,\boldsymbol{x}_2) & \dots & K(\boldsymbol{x}_m,\boldsymbol{x}_m) \end{pmatrix}. \tag{3.3.16}$$

Solche Matrizen sind ihrerseits ebenfalls symmetrisch und positiv semidefinit. Eine Matrix G ist dabei genau dann positiv semidefinit, falls

$$\boldsymbol{x}^\top G\boldsymbol{x} \geq 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^n$$

gilt. Die Auffassung eines Kernoperators als Ähnlichkeitsmaß, das Werte größer oder gleich Null annimmt, ist ein Spezialfall dieser Formulierung. Allgemein können Skalarprodukte, und damit die Einträge der Gram-Matrix, auch negativ sein.

Für den Fall, dass alle Kernoperatorwerte $K(\boldsymbol{x}, \boldsymbol{y})$ zwischen allen Objekten $\boldsymbol{x}$ und $\boldsymbol{y}$, die für Training und Klassifikation in Betracht gezogen werden, bekannt sind, benötigt die SVM die zugrunde liegenden Objekte gar nicht mehr. Dass auch das Training von den eigentlichen Objekten unabhängig ist, sieht man durch die Betrachtung des dualen Optimierungsproblems für die Hyperebene $\mathcal{H}$ aus Gleichung 3.3.11. Auch hier gehen die Objekte nur als Skalarprodukt ein. Durch diese allgemeine Repräsentation der Objekte von Interesse und die Beziehung zwischen ihnen wird eine ganze Klasse von Methoden möglich, von denen im Rahmen dieser Arbeit einige verwendet werden.

Diese Definitionen von Kernoperatoren und Angaben ihrer Eigenschaften gehören zu einem Methodikrepertoire, das oft in praktischen Anwendungen angetroffen wird. Es soll noch eine weitere Möglichkeit aufgeführt werden, gültige Kernwerte zu erhalten, die in Abschnitt 4.7 verwendet wird. Séaghdha und Copestake (2008) stellen den Begriff des *negativ semidefiniten Kernoperators* $\hat{K}$ vor. Die Funktion $\hat{K} : \boldsymbol{\Omega} \times \boldsymbol{\Omega} \to \mathbb{R}$ sei symmetrisch und für die durch $\hat{K}$ induzierte Gram-Matrix G gelte

$$\boldsymbol{x}^\top G\boldsymbol{x} \leq 0, \quad \forall \boldsymbol{x} \in \mathbb{R}.$$

Es lassen sich dann auf zwei Arten und Weisen gültige, positiv semidefinite Kernoperatoren K aus $\hat{K}$ gewinnen:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \hat{K}(\boldsymbol{x}, \boldsymbol{x}_0) + \hat{K}(\boldsymbol{y}, \boldsymbol{x}_0) - \hat{K}(\boldsymbol{x}, \boldsymbol{y}) - \hat{K}(\boldsymbol{x}_0, \boldsymbol{x}_0), \quad \forall \boldsymbol{x}_0 \in \boldsymbol{\Omega} \tag{3.3.17}$$

$$K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\alpha \hat{K}(x,y)}, \quad \forall \alpha > 0 \tag{3.3.18}$$

Dies entspricht der Konversion eines Distanzmaßes in ein Ähnlichkeitsmaß. Der Punkt $\boldsymbol{x}_0$ kann als Referenz eines Koordinaten-Ursprungs für die Transformation im Raum $\mathbb{H}$ gesehen werden. Seine Wahl wirkt sich nach Séaghdha und Copestake (2008) nicht auf die SVM-Klassifikation aus. Da die Gleichung 3.3.17 eine lineare Transformation darstellt, werden auf diese Weise gewonnene Kernoperatoren auch als *linear transformierte Kerne*, in Abgrenzung zum linearen Kern $\langle \cdot, \cdot \rangle$, bezeichnet. Gleichung 3.3.18 formt

hingegen durch die Symmetrie des Operators $\hat{K}$ gaußähnliche Funktionen. Zudem ist die Operatordefinition ähnlich zur Definition des RBF-Kerns aus Gleichung 3.3.13. Deshalb werden so gewonnene Kernoperatoren im Folgenden mit dem Namenszusatz „RBF“ referenziert.

KAPITEL 4

Ausgewählte Ansätze für die Relationsextraktion

In den letzten Jahren wurden zahlreiche Methoden entwickelt, um das Problem der PPI-Extraktion zu lösen. Viele Ansätze bedienen sich der SVM-Klassifikation wegen der guten Verallgemeinerung, die SVMs zugeschrieben wird, sowie der Möglichkeit, Kernwerte auf beliebigen Objekten zu berechnen. Für diese Arbeit wurde eine Auswahl unter diesen Techniken getroffen, die reimplementiert wurde. Die Auswahlkriterien richteten sich dabei sowohl nach den Ergebnissen, die die Autoren für das jeweilige Verfahren berichteten, als auch nach der Abdeckung eines größeren Feldes von Lösungsansätzen. Durch ein - gemessen am Umfang dieser Arbeit - großes Spektrum von verschiedenen Strategien soll eine Grundlage für Vergleiche sowie die bereits in Kapitel 2 erwähnte und mit Gleichung 3.3.14 erläuterte Kombination von verschiedenen Kernoperatoren erreicht werden. Zudem wird auf diese Weise eine Bibliothek von Methoden errichtet, aus der passend für eine konkrete Anwendung die beste Technik ausgewählt werden kann. Im Folgenden werden die ausgewählten Techniken vorgestellt und erläutert. Dabei geht es an dieser Stelle um die prinzipiellen Funktionsweisen und nicht um Details der Implementierung, die in Kapitel 7 besprochen werden. Es wird zwischen Ansätzen unterschieden, die eine Merkmalrepräsentation verwenden und Ansätzen, die direkt auf dem zu Grunde liegenden Objekt einen Kernwert berechnen. Die Merkmalansätze werden zuerst beschrieben. Außerdem werden die Verfahren innerhalb ihrer Kategorie nach wachsendem Gebrauch von Informationen des Textes geordnet, aus dem die Interaktionen extrahiert werden sollen. Bevor die Methoden vorgestellt werden, werden grundlegende Betrachtungen bezüglich der Vorgehensweise mit einem Merkmal- bzw. Kernansatz sowie ein Vergleich der prinzipiellen Methodiken gegeben.

4.1 Grundlegende Vorgehensweise

Es soll zum besseren Verständnis noch einmal auf die genaue Verwendungsweise von Merkmal- bzw. Kernansätzen eingegangen werden. Merkmalansätze stellen Verfahren dar, die bestimmte Merkmale extrahieren und auf diese Weise einen Zahlenvektor für jeden PPI-Kandidaten erstellen. Als *Eingabe* für ein solches Verfahren dient *ein Satz*, in dem die beiden möglichen Relationsargumente markiert sind. Es wird im Folgenden öfter die Formulierung verwendet, dass bestimmte Eigenschaften von Wörtern, Sätzen, Syntaxbäumen etc. als Merkmal betrachtet werden. Damit ist stets gemeint, dass jede konkrete Ausprägung dieser Eigenschaften eine eigene Position - und damit Dimension - im resultierenden Merkmalsvektor erhält und - soweit nicht ausdrücklich anders erklärt - auf den Wert Eins gesetzt wird. Die resultierenden Merkmalsvektoren sind demnach in den meisten Fällen Binärvektoren. Dieses Vorgehen wurde bereits in Abschnitt 3.2 beschrieben. Ein konkreter Merkmalsvektor ist folglich von sehr hoher Dimension und äußerst dünn besetzt, da die meisten konkreten Eigenschaftsausprägungen in einem einzelnen Satz nicht aufzufinden sind. So gewonnene Vektoren $\boldsymbol{x}$ und $\boldsymbol{y}$ zweier Objekte können dann beispielsweise mit einem RBF-Kern verwendet werden, wie er in Gleichung 3.3.13 beschrieben wurde. Auf Basis dieser Kernwerte ist eine SVM in der Lage, eine separierende Hyperebene im hochdimensionalen Raum $\mathbb{H}$ zu finden und neue Objekte gemäß dieser Ebene zu klassifizieren.

Die Kernoperatoren, die in diesem Kapitel beschrieben werden, arbeiten nicht wie der RBF-Kern auf Vektoren. Sie erhalten *als Eingabe zwei Sätze*, die jeweils eine potentielle PPI enthalten, deren Argumente markiert sind. Aus diesen Sätzen werden voneinander unabhängig nicht-vektorielle Darstellungen generiert, auf deren Grundlage anschließend direkt der Kernwert der beiden PPI-Kandidaten berechnet wird.

4.2 Merkmaldarstellung vs. direkte Objektabbildung

Bevor die Verfahren beschrieben werden, sollen einige grundsätzliche Gedanken bezüglich der Objektdarstellung einander gegenübergestellt werden. Obwohl die Kerntechnik nicht neu ist, gehören Merkmalansätze nach wie vor zum regulären Repertoire von Algorithmen des maschinellen Lernens (engl. Maschine Learning), auch wenn sie zusammen mit dem Kernel-Trick einsetzbar sind. Der Einsatz von Merkmal- bzw. Featurefunktionen versetzt den Autor einer Machine Learning (ML)-Anwendung in die Situation, mit einfachen Mitteln Informationen über die zu verarbeitenden Objekte zusammentragen zu können. Durch den Einsatz einer Reihe voneinander unabhängiger Merkmalfunktionen ist es außerdem leicht möglich, eine beliebige Teilmenge der vor-

handenen Merkmale zu verwenden, also einzelne Funktionen innerhalb einer Anwendung zu aktivieren oder zu deaktivieren. Auf diese Weise können Suchtechniken eingesetzt werden, die sich der Auffindung guter Merkmalmengen widmen, die so genannte Feature Subset Selection. Eine gute Merkmalmenge definiert sich dabei generell dadurch, dass ein mit ihnen angelernter Klassifikator eine hohe Erkennungsrate aufweist. Durch diese Unabhängigkeit der Merkmale voneinander[1] ist es jedoch schwierig, Strukturinformationen durch Merkmale zu kodieren. So gibt es etwa keinen allgemein erfolgreichen Weg, die Struktur eines Baums oder gar Graphen durch Merkmalfunktionen darzustellen. An dieser Stelle fehlen Mittel wie etwa die exakte mathematische Formulierung der Objektrepräsentation, um zu beurteilen, wie gut oder wie „natürlich“ ein Objekt durch eine feste Menge von Merkmalen dargestellt wird. Aber gerade durch den Einsatz von Parsinginformationen in der PPI-Extraktion wird es zunehmend wichtig, solche Informationen in das Lernverfahren mit einzubeziehen.

An dieser Stelle bieten Kernverfahren einen potentiellen Ausweg, weil sie nicht zwingend auf eine Repräsentation durch Merkmale zurückgreifen müssen. In Unterabschnitt 3.3.3 wurden Kernoperatoren vorgestellt, die auf einer bereits vorhandenen Vektorrepräsentation von Objekten operieren - also mit einer Merkmaldarstellung. Es wurde ebenfalls beschrieben, dass beliebige Funktionen $K(\cdot, \cdot)$ als Kernoperator verwendet werden können, so lange sie die Eigenschaften der Symmetrie und positiven Semidefinitheit besitzen. Der Wert des Kerns wird dann als Ähnlichkeitswert interpretiert: Je höher der Kernwert zwischen zwei Objekten, desto höher ihre Ähnlichkeit. Damit wird es möglich, Kernwerte von beliebigen Objekten zu erhalten. Es können auf diese Weise Bäume („Tree Kernel“) und sogar Graphen („Graph Kernel“) miteinander verglichen werden, wodurch die syntaktische Struktur von Sätzen adäquat erfasst werden kann. Im Falle der PPI-Extraktion ist im Wesentlichen von Belang, wie der (Syntax-)Baum bzw. Graph dargestellt wird und welche Informationen ihm zugrunde liegen. Tree Kernels werden oft eingesetzt, um direkt die Ähnlichkeit zwischen zwei (Teil-)Ableitungsbäumen zu berechnen. Für Graph-Kerne muss eine bestimmte, den mathematischen Grundlagen des Kernoperators entsprechende, Darstellung eines Graphen geliefert werden. Diese basiert sinnvollerweise zwar auf dem Ableitungsbaum (oder sogar Ableitungsgraph, wie in Abschnitt 3.1 erwähnt wurde), muss ihr aber nicht völlig entsprechen. An dieser Stelle muss wieder an einem bestmöglichen Transport von Information vom zugrunde liegenden Objekt zu seiner für den Kernoperator K verarbeitbaren Form gearbeitet werden.

Es scheint daher, als wäre die direkte Berechnung von Kernwerten der Verwen-

[1] In der Tat kommt es natürlich vor, dass zwei Merkmalfunktionen f_i und f_j keineswegs statistisch unabhängig voneinander sind. Hier ist die unabhängige Einsetzbarkeit der Merkmale gemeint.

dung von Merkmalen potentiell überlegen. Ohne den Umweg über die Merkmaldarstellung kann der Informationsfluss differenzierter gestaltet werden und es stehen größere Freiheiten in der Modellierung eines konkreten Algorithmus' zur Verfügung. Für diese Intuition ist jedoch keine verlässliche oder gar beweisbare Grundlage bekannt. Deshalb wurden Methoden aus beiden Techniken ausgewählt, die später miteinander verglichen werden sollen.

4.3 Bag-of-Words-Merkmale

Die hier beschriebenen Merkmale dienen nicht primär als eigenständige Methode für die PPI-Extraktion. Es wird später noch deutlich werden, dass für die bereits durch Gleichung 3.3.14 beschriebene Kombination nicht alle möglichen Paare von vorhandenen Methoden in Frage kommen. Diese Problematik wird in Kapitel 7 und Kapitel 8 ausgeführt. Die Bag-of-Words-Merkmale sollen auf Grund ihrer Simplizität als sicherer Kombinationspartner dienen und werden deshalb eingeführt. Die hier vorgestellten Merkmale wurden speziell für diese Aufgabe entwickelt und folgen in ihrem Konzept keiner speziellen externen Quelle.

Ein *Bag-of-Words* bezeichnet eine ungeordnete Menge von Wörtern, d.h. es liegt keine Sequenzinformation vor. Hier werden alle Wörter betrachtet, die in einem Satz vorkommen, aus dem die zwei potentiellen Relationsargumente gegeben sind. Dabei erhalten die Argumente sowie alle Wörter, die zwischen ihnen stehen, ein Merkmalgewicht von Zwei. Alle anderen Wörter werden mit Eins gewichtet.

Durch die einfach gehaltene Methodik unterliegen diese Merkmale keiner Beschränkung und können stets verwendet werden. Diese Eigenschaft steht beispielsweise im Gegensatz zu den Merkmalen, die in Abschnitt 4.5 beschrieben werden. Dort muss eine Dependenzanalyse eines betrachteten Satzes vorliegen und ein kürzester Dependenzpfad zwischen den potentiellen Argumenten existieren. Anderenfalls können die Merkmale nicht erstellt werden.

4.4 Merkmale bezüglich Dependenzbaumebenen

(Katrenko und Adriaans 2007)

In diesem Abschnitt wird eine Menge von Merkmalen beschrieben, die von Katrenko und Adriaans (2007) vorgeschlagen wurden. Diese Merkmale beschreiben die Eigenschaften des Dependenzbaumes, in dem zwei potentielle PPI-Argumente X und Y enthalten sind, durch ebenenbezogene Informationen des Baums relativ zu X und Y. Die Argumente X und Y sind folglich NEs, die durch die Vorprozessierung markiert

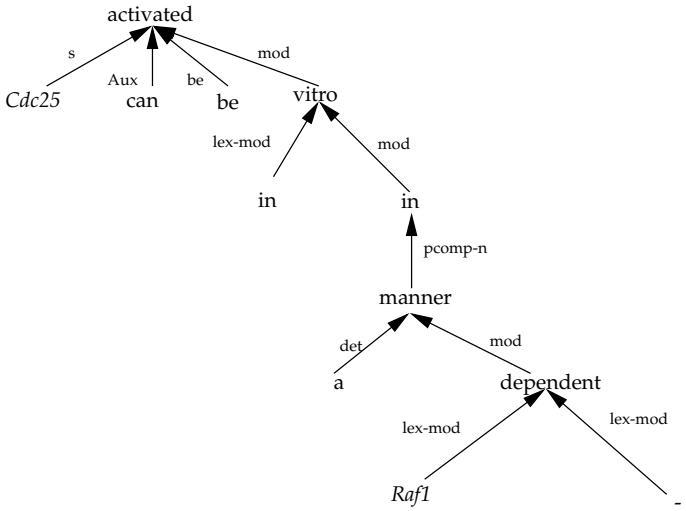

Abbildung 4.1: Der Beispielsatz „*Cdc25 can be activated in vitro in a Raf1-dependent manner*“ aus dem AIMed-Corpus. Die potentiellen Argumente *Cdc25* und *Raf1* sind kursiv dargestellt.

wurden. Die Merkmale, die im Folgenden beschrieben werden, lassen sich in zwei Kategorien einteilen: In lokale Merkmale und globale Merkmale. Mit lokalen Merkmalen sind Eigenschaften des Dependenzbaumes gemeint, die in einem örtlich begrenzten Zusammenhang mit einem der Argumente X oder Y stehen. Globale Merkmale hingegen beziehen sich allgemeiner auf den Dependenzbaum des Satzes, in dem sich die potentiellen Argumente befinden. Das bedeutet allerdings nicht, dass die letzteren Merkmale für alle möglichen Argumentenpaare im Satz gleich sein müssen, da auch globale Merkmale von den konkreten Argumentkandidaten abhängen können.

Um die Erstellung der Merkmale zu verdeutlichen, soll ein Beispielsatz aus dem AIMed-Corpus (vgl. Kapitel 6) zur Illustration dienen. Er ist in Abbildung 4.1 zu sehen. Als lokaler Kontext eines potentiellen Arguments X werden der Vaterknoten von X (bezeichnet mit P) sowie zwei seiner Kinderknoten (C^1 und C^2) im Dependenzbaum betrachtet. Es ist zwar möglich, dass ein Knoten mehr als zwei Kinder besitzt, diese gehen aber nicht in die Merkmalrepräsentation ein. Da nicht a priori bekannt ist, wie viele Kinder es für einen Knoten gibt, lassen sich keine verlässlichen Merkmalwerte erhalten, wenn mehr als zwei Kinder betrachtet werden. Die Information der Dependenzkante, die einen Knoten mit seinem Vater bzw. einem Kind verbindet, also die Kantenbeschriftung, wird in die Merkmale für die Vater- und Kinderknoten integriert, wie weiter unten anhand eines Beispiels gezeigt wird.

Als globaler Kontext zweier Kandidaten X und Y gelten zum Einen die Wurzel des Knotens (R) und zum Anderen der niedrigste gemeinsame Vorgänger (engl. Least Common Subsumer (LCS)) der Knoten X und Y im Dependenzbaum. Für jedes Paar

Merkmal	*Cdc25*	*Raf1*
C^1	-	-
C^1	-	-
P	activate_s	dependent_lexmod
LCS	activate	activate
R	activate	activate

Tabelle 4.1: Merkmale des Beispielbaums 4.1. Die Wörter wurden durch Lemmatisierung normalisiert, die Dependenzbeschriftung für Vater und Kinder ist durch einen Unterstrich mit dem Lemma verbunden angegeben.

von Knoten in einem Baum existiert der LCS und ist eindeutig bestimmbar.

Zur Verdeutlichung soll ein kurzes Beispiel anhand Abbildung 4.1 folgen. Als X bzw. Y seien die Knoten *Cdc25* und *Raf1* gegeben. Ihr LCS stellt der Knoten *activated* dar, der zugleich als Wurzel des Baums dient. *Cdc25* hat *activated* als Vaternoten und ist durch eine Dependenzkante mit der Bezeichnung *s* mit ihm verbunden. Kinder besitzt *Cdc25* nicht. *Raf1* hat den Vater *dependent* mit der Kantenbezeichnung *lexmod*, Kinder besitzt er ebenfalls nicht. Diese Merkmale sind in Tabelle 4.1 zusammengefasst. Es ist zu sehen, dass nicht die vollen Wortformen als Merkmale verwendet werden. Statt dessen wird das Lemma eines Wortes, also dessen Grundform, eingesetzt, um die Datendichte so hoch wie möglich zu halten. Kleine Unterschiede wie Flexionen sind im Kontext der PPI-Extraktion von keinem größeren Belang, so dass sie eher als Rauschen in den Daten wirken können. Deshalb wird an dieser Stelle das Lemma als Wortnormalisierung verwendet.

Im restlichen Teil der Arbeit werden die in diesem Abschnitt vorgestellten Merkmale auch verkürzend als *Dependenzmerkmale* bezeichnet.

4.5 Walk-Merkmale

(Kim u. a. 2008)

Kim u. a. (2008) schlagen eine Reihe von Merkmalen und direkten Kernoperatoren vor. Das Verfahren, das ihrem Bericht zufolge die besten Resultate erzielt, verwendet die so genannten Walk-Merkmale. Das Gesamtverfahren wird von den Autoren als *Walk-Kernel* bezeichnet, obwohl es sich um eine Merkmalmethode handelt. Sie wird mit einem Polynomialkern verwendet, was aber keinen wesentlichen Bestandteil des Verfahrens an sich darstellt.

Das Verfahren beruht auf einem Grundprinzip, das von Bunescu und Mooney (2005) entwickelt wurde. Die als *Shortest Path Hypothesis* - also *Hypothese des kürzesten*

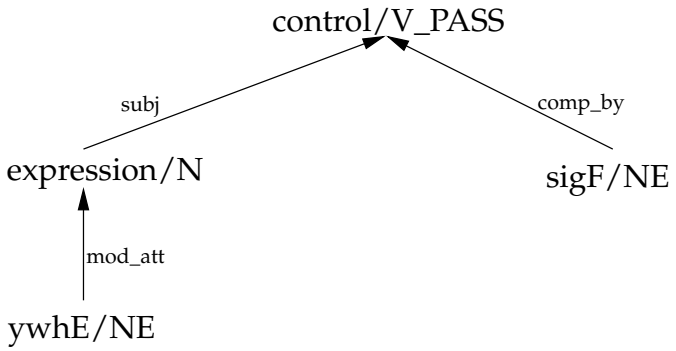

Abbildung 4.2: Der kürzeste Pfad zwischen den Entitäten ywhE und sigF im Beispielsatz '*Analysis of the expression of a translational ywhE-lacZ fusion showed that ywhE expression is sporulation-specific, and is controlled predominantly by the forespore-specific sigma factor sigma(F), and to a lesser extent by sigma(G).*' aus dem LLL05-Corpus. Für das Auffinden des Pfades dürfen die (gerichteten) Kanten in beide Richtungen traversiert werden.

Pfades - formalisierte Idee beruht auf der Annahme, dass die wichtigsten Informationen im Dependenzgraphen, um eine Relation zwischen zwei Entitäten e_1 und e_2 in einem Text auszudrücken, mit dem kürzesten ungerichteten Dependenzpfad zwischen e_1 und e_2 gegeben sind. Sind e_1 und e_2 Teile einer Prädikat-Argument-Struktur, wie beispielsweise 'e_1 *activates* e_2', liegt das für die Relationserkennung wichtige Prädikat ebenfalls auf dem Pfad. Bei komplizierteren Strukturen, wie etwa verschachtelte Prädikat-Argument-Strukturen, die eine der Entitäten e_1 oder e_2 als Argument teilen, wären sogar beide Prädikate im Pfad enthalten. Zusätzlich werden Konjunktionen, die ebenfalls hilfreich sein können, um Beziehungen auszudrücken, berücksichtigt.

Der kürzeste Pfad zwischen zwei Entitäten ist von variabler und von der Textstruktur abhängiger Länge. Potentiell enthält er mehr Informationen, wenn der zugrunde liegende Text mehr Informationen liefern kann. Diese Eigenschaft steht etwa im Gegensatz zu der Art und Weise der Informationskodierung im Falle der einfachen Dependenzmerkmale aus Abschnitt 4.4, die nur feste Bezugspunkte im Dependenzgraphen berücksichtigen.

Um nun eine Merkmalkodierung des kürzesten Pfades zwischen zwei Entitäten zu erhalten, schlagen Kim u. a. (2008) die Erstellung von *Walks* vor. Damit sind kurze Abschnitte des Pfades gemeint, die im Folgenden mathematisch gefasst werden sollen.

Sei der Dependenzgraph eines Satzes durch $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ gegeben, wobei $\mathcal{V} = \{v_1, \ldots, v_n\}$ die Knoten- und $\mathcal{E} = \{e_1, \ldots, e_m\}$ die Kantenmenge des Graphen bezeichne. Ein Pfad in $\mathcal{G}$ der Länge l ist eine Folge von Knoten $v_i \in \mathcal{V}$ und Kanten $e_{i,j} \in \mathcal{E}$ der Gestalt $\mathcal{P} = v_i, e_{i,i+1}, v_{i+1}, \ldots, e_{i+l-2,i+l-1}, v_{i+l-1}$, die mit einem Knoten

Lexikalische Walks	NE+mod_att(UP)+expression, mod_att(UP)+expression+subj(UP), expression+subj(UP)+control, subj(UP)+control+comp_by(DN), control+comp_by(DN)+NE
Syntaktische Walks	NE+mod_att(UP)+N, mod_att(UP)+N+subj(UP), N+subj(UP)+V_PASS_PRED, subj(UP)+V_PASS_PRED+comp_by(DN), V_PASS_PRED+comp_by(DN)+NE

Tabelle 4.2: Walk-Merkmale für das Beispiel 4.2.

beginnt und aufhört. Ein Walk auf einem Pfad $\mathcal{P}$ sei definiert als beliebige Sequenz von drei in $\mathcal{P}$ direkt aufeinander folgenden Elementen. Das heißt, es werden sowohl Sequenzen $v_i, e_{i,i+1}, v_{i+1}$ als auch $e_{i-1,i}, v_i, e_{i,i+1}$ betrachtet. Zusätzlich wird zwischen *lexikalischen Walks* und *syntaktischen Walks* unterschieden. Lexikalische Pfadabschnitte verwenden als Knotenbeschriftung das Wort bzw. Token, das dem korrespondierenden Graphknoten entspricht. Ein syntaktischer Walk verwendet stattdessen die Wortart des Tokens. Beide Darstellungen verwenden die Kantenbeschriftung, um eine Kante für die Merkmale darzustellen. Die Kanten erhalten eine zusätzliche Beschriftung „(UP)“ oder „(DN)“ je nachdem, ob sich im Dependenzgraphen hinauf oder hinunter bewegt wird, um Strukturinformationen des Pfades einzubringen. Als zusätzlich beschreibende Information wird die Zeichenkette „PRED“ dem Prädikat angehängt. Die Entitäten werden generell durch die Zeichenkette „NE“ dargestellt, um eine höhere Datendichte zu erhalten. Es werden außerdem, wie im vorigen Abschnitt, Lemmata[2] anstatt der Vollformen der Wörter verwendet. In Abbildung 4.2 wird ein kürzester Pfad zwischen zwei Entitäten in einem Beispielsatz gezeigt. Tabelle 4.2 zeigt die konkreten Merkmalausprägungen bezüglich dieses Pfades.

4.6 Vereinigung von Merkmalmengen

In Abschnitt 4.2 wurde ausgeführt, dass ein Vorteil der Verwendung von Merkmalen in der Möglichkeit besteht, sie gezielt ein- oder auszuschalten. Durch diese Eigenschaft ist es ebenso möglich, unterschiedliche Merkmalmengen zu vereinigen. In dieser Arbeit wird die Vereinigung der Merkmale aus Abschnitt 4.4 sowie Abschnitt 4.5 mit einer

[2] Als Lemma wird die Grundform eines Wortes bezeichnet.

weiteren Menge von Merkmalen in Betracht gezogen. Diese letzte Menge soll nur zu Kombinationszwecken dienen, weshalb sie in diesem Abschnitt beschrieben wird.

Es handelt sich bei den zusätzlich verwendeten Merkmalen um eine Reihe von Eigenschaften, die großteils ursprünglich von GuoDong u. a. (2005) für die Relationsextraktion auf Zeitungstexten vorgeschlagen wurden. Sie sind in die Kategorien Wort-Merkmale, Merkmale bezüglich Basisphrasen-Chunking[3] und Parse-Merkmale Alle Kategorien wurden außerdem mit leichten Modifikationen von Buyko u. a. (2008) verwendet und beschrieben. Diese Modifikationen wurden vorgenommen, um die Merkmale an die biomedizinische Domäne anzupassen. Daher wird auch in dieser Arbeit mit den modifizierten Merkmalmengen gearbeitet.

Bei den **Wort-Merkmalen** handelt es sich um Eigenschaften des Satzes bezüglich der Argumentkandidaten auf Wortebene. Es werden etwa Wörter vor der ersten Entität betrachtet, die einen Argumentkandidaten darstellt, sowie Wörter nach der zweiten Entität. Falls nur ein Wort zwischen den Entitäten steht, so wird auch dieses als Merkmal betrachtet. Eine vollständige Liste der Wortmerkmale ist in Abschnitt A.1 gegeben.

Die **Merkmale bezüglich Basisphrasen-Chunks** folgen vom Muster her den Wort-Merkmalen. Das heißt, es werden Phrasen vor, zwischen und nach den potentiellen Relationsargumenten betrachtet. Dabei werden immer die Köpfe der Chunks als konkretes Merkmal verwendet, also zumeist das letzte Wort einer Phrase. Die genauen Merkmale sind in Abschnitt A.2 aufgeführt.

Für die **Parse-Merkmale** werden die Konstituenten[4] eines Strukturbaums des betrachteten Satzes herangezogen. Dabei werden einzelne Konstituenten, im speziellen Nominal[5]-, Präpositional[6]- und Verbalphrasen[7], daraufhin untersucht, ob sie beide potentiellen Argumente enthalten. Zudem wird der Pfad im Strukturbaum betrachtet, der die Argumente miteinander verbindet. In Abschnitt A.3 werden die Merkmale mit Beschreibung aufgelistet.

Durch die Konfigurierbarkeit und das einheitliche Erscheinungsbild der Applikation, die zur Realisierung der in diesem Kapitel beschriebenen Methoden dient (vgl. Abschnitt 7.1), können alle diese Merkmale leicht zusammengeführt werden.

[3] Bei Basisphrasen-Chunks handelt es sich um Wortgruppen, die über keine innere Struktur, wie etwa weitere Chunks, verfügen.

[4] Satzglieder; Wortgruppen, die innerhalb des Satzes zumeist nur gemeinsam verschoben werden können.

[5] Eine Phrase, deren Kopf ein Substantiv ist.

[6] Eine Phrase, deren Kopf eine Präposition ist.

[7] Eine Phrase, deren Kopf ein Verb ist.

4.7 Verteilungskern

(Séaghdha und Copestake 2008)

Die Grundidee von Séaghdha und Copestake (2008) besteht in der Verwendung von Wahrscheinlichkeitsverteilungen bezüglich der potentiellen Relationsargumente, um das Problem der Relationserkennung zu lösen. Dazu werden Maße verwendet, die dem Vergleich von unterschiedlichen Wahrscheinlichkeitsverteilungen dienen. Um diese schließlich mit einer SVM verwenden zu können, werden von diesen Vergleichsmaßen Kernoperatoren nach den Formeln 3.3.17 und 3.3.18 abgeleitet.

Im Konkreten handelt es sich bei den Wahrscheinlichkeitsverteilungen um Kookkurrenzmaße, also um die Wahrscheinlichkeit für zwei Tokens t_1 und t_2, in einer bestimmten Relation r zueinander zu stehen. Die Relation r hat dabei typischerweise etwas mit textueller Nähe zu tun und besteht in der einfachsten Form aus der Nachbarschaftsrelation. Zwei Tokens werden demnach als kookkurrent aufgefasst, wenn sich beide innerhalb eines Bereiches von beispielsweise drei zusammenhängenden Tokens in einem Text befinden. Durch die Annahme, dass die zu verarbeitenden Texte mit einem Dependenzparser analysiert wurden, ergeben sich jedoch auch weitere Möglichkeiten. Es ist dadurch möglich zu bestimmen, ob zwei Tokens in einer bestimmten syntaktischen Relation zueinander stehen, wie es bei Verb-Argument- oder Konjunktion-Konjunktion-Relationen der Fall ist. Um eine Wahrscheinlichkeitsverteilung über Kookkurrenzen erstellen zu können, ist es nötig, sich auf eine Grundgesamtheit bzw. ein endliches Vokabular $\mathcal{V}$ festzulegen. Aus Notationsgründen werden die betrachtete Relation sowie die in Frage kommenden Kookkurrenztokens aus $\mathcal{V}$ zu *Kookkurrenztypen* $c_i \stackrel{def}{=} (r, t_i)$ zusammengefasst. $P(c_i|t_j)$ bezeichne dann die Wahrscheinlichkeit, dass die Relation r für t_i gelte, wenn t_j als zweites Argument von r gegeben ist. Die Menge der Relationstypen sei als $\mathcal{C}$ bezeichnet. Ein Token $t_i \in \mathcal{V}$ wird dann durch einen Vektor von Kookkurrenzwahrscheinlichkeiten zu allen anderen Tokens in $\mathcal{V}$ dargestellt:

$$t_i^r \stackrel{def}{=} (P(c_1|t_i), P(c_2|t_i), \ldots, P(c_{|\mathcal{C}|}|t_i))^\top, \quad \forall t_i \in \mathcal{V}, \forall c_i \in \mathcal{C},$$

wobei r eine Kookkurrenzrelation zwischen zwei Tokens t_i und t_j bezeichne. Da die t_i^r Wahrscheinlichkeitsverteilungen darstellen, gilt $\sum_{k=1}^{|\mathcal{C}|}(t_i^r)_k = 1, \quad \forall i \in \{1, \ldots, |\mathcal{C}|\}$.

Sind solche Verteilungen gegeben, muss für die Verwendung mit einer SVM ein Kernoperator formuliert werden, so dass diese Informationen zur Klassifikation verwendet werden können. Mögliche Maße, um zwei Wahrscheinlichkeitsverteilungen miteinander zu vergleichen, sind durch die Informationstheorie in großer Zahl gegeben und bezeichnen typischerweise Distanzmaße, die bei hoher Abweichung der ver-

Abstandsmaß	Definition	Abgeleiteter linearer Kern
L_1-Abstand	$\sum_{c \in \mathcal{C}} \lvert P(c\lvert t_i) - P(c\lvert t_j) \rvert$	$\sum_{c \in \mathcal{C}} \min(P(c\lvert t_i), P(c\lvert t_j))$
$(L_2$-Abstand$)^2$	$\sum_{c \in \mathcal{C}} (P(c\lvert t_i) - P(c\lvert t_j))^2$	$\sum_{c \in \mathcal{C}} P(c\lvert t_i) P(c\lvert t_j)$
Jensen-Shannon-Divergenz	$\sum_{c \in \mathcal{C}} P(c\lvert t_i) \log_2(\frac{2P(c\lvert t_i)}{P(c\lvert t_i) + P(c\lvert t_j)}) + P(c\lvert t_j) \log_2(\frac{2P(c\lvert t_j)}{P(c\lvert t_i) + P(c\lvert t_j)})$	$-\sum_{c \in \mathcal{C}} P(c\lvert t_i) \log_2(\frac{P(c\lvert t_i)}{P(c\lvert t_i) + P(c\lvert t_j)}) + \sum_{c \in \mathcal{C}} P(c\lvert t_j) \log_2(\frac{P(c\lvert t_j)}{P(c\lvert t_i) + P(c\lvert t_j)})$
Hellinger-Abstand	$\sum_{c \in \mathcal{C}} (\sqrt{P(c\lvert t_i)} - \sqrt{P(c\lvert t_j)})^2$	$\sum_{c \in \mathcal{C}} \sqrt{P(c\lvert t_i) P(c\lvert t_j)}$

Tabelle 4.3: Klassische Verteilungsmaße aus der Informationstheorie und die daraus abgeleiteten linearen Kernoperatoren.

glichenen Verteilungen große Werte annehmen. Da für die Ableitung von positiv semidefiniten Kernoperatoren durch die Formeln 3.3.17 und 3.3.18 symmetrische, negativ semidefinite Kerne als Ausgangspunkt nötig sind, fallen nichtsymmetrische Maße aus der Betrachtung. Séaghdha und Copestake (2008) schlagen als Maße den *L_1-Abstand*, *L_2-Abstand*, die *Jensen-Shannon-Divergenz* sowie den *Hellinger-Abstand* vor. Die Kerne, die durch lineare Transformation gewonnen wurden, im Folgenden einfach *lineare Kerne* genannt (vgl. Unterabschnitt 3.3.3), werden zusammen mit der Definition der ihnen zugrunde liegenden Distanzmaße beispielhaft in Tabelle 4.3 aufgeführt. Es wird an dieser Stelle darauf hingewiesen, dass der L2-normierte, lineare L2-Kern gerade dem Cosinus-Ähnlichkeitsmaß entspricht (Séaghdha und Copestake 2008). Dieses Maß wird im Zuge des nächsten Abschnitts in Tabelle Tabelle 4.4 vorgestellt.

Um mit diesen Mitteln nun Kernwerte für die Relationsextraktion zu erhalten, wird wie folgt vorgegangen: Als Tokens t_i und t_j dienen die beiden Argumente einer potentiellen Textrelation, die zu extrahieren ist. Die Vektoren t_i^r und t_j^r der Kookkurrenzverteilungen der beiden Tokens werden konkateniert, so dass ein einzelner langer Vektor entsteht. Dieser wird um den Faktor 0.5 skaliert, um die Verteilungseigenschaft wieder herzustellen. Da ein Kernoperator K die Ähnlichkeit zweier Objekte misst, wobei die Objekte in diesem Fall mögliche Relationen zwischen Entitäten darstellen, wird dieses Verfahren für alle Entitätenpaare im zu prozessierenden Text durchgeführt. Ein Kernwert zweier Relationen ergibt sich dann durch die Berechnung eines Ähnlichkeitswertes zweier Vektoren, die wie oben beschrieben durch Konkatenation konstruiert wurden und damit jeweils einer potentiellen Relation entsprechen.

Zwei Objekte - potentielle Relationen - $\boldsymbol{x}$ und $\boldsymbol{y}$ sind sich demnach ähnlich, falls

ihre Argumente eine ähnliche Kookkurrenzverteilung besitzen oder, bildlicher gesprochen, falls ihre Argumente aus statistischer Sicht zu einem bestimmten Grad gegeneinander austauschbar sind. Es wird damit die Idee realisiert, dass Relationen durch Art und Häufigkeit anderer Wörter bzw. Tokens in der Umgebung der Argumentkandidaten charakterisierbar sind. Die Technik wurde ursprünglich basierend auf allgemeinsprachlichen Daten entwickelt und stellt keine spezielle Methode für die PPI-Extraktion dar. Ein Problem der Portierung in den biomedizinischen Bereich könnte die hohe Diversität der Bezeichnung von Genen und Proteinen sowie die große Anzahl dieser Entitäten darstellen. Das könnte dazu führen, dass für viele Entitäten keine bzw. keine repräsentative Wahrscheinlichkeitsverteilung vorliegt. Diese Vermutung wird im Auswertungsteil der Arbeit näher untersucht.

4.8 Local-Alignment-Kern

(Katrenko und Adriaans 2008)

Katrenko und Adriaans (2008) schlagen die Berechnung von Kernwerten auf Grund eines Vergleichs von Zeichenketten vor, die kürzeste Dependenzpfade zwischen potentiellen Relationsargumenten repräsentieren. Ähnlich wie in Abschnitt 4.5 liegt dem Gesamtverfahren die Hypothese des kürzesten Pfades von Bunescu und Mooney (2005) zugrunde. Hier handelt es sich jedoch um keine Merkmalextraktion sondern um die Definitition eines Kernoperators, der als Eingabe zwei Pfade erhält und einen Ähnlichkeitswert auf ihnen berechnet. Als zugrunde liegendes Maß wird das *Smith-Waterman-Ähnlichkeitsmaß* (Smith und Waterman 1981) verwendet, das in seiner ursprünglichen Version zum Vergleich von Molekularsequenzen entwickelt wurde. Bei diesem Maß handelt es sich um eine Methode zum Vergleich von Zeichenketten, ähnlich dem Levenshtein-Abstand. Der Levenshtein-Abstand berechnet ein *global Alignment*, also eine global optimale Ausrichtung von symbolischen Elementen zueinander, unter Einbeziehung von Strafpunkten für bestimmte Operationen, die eine Zeichenkette in die andere überführen. Erlaubte Operationen sind das Einfügen, das Löschen und der Austausch von Zeichen mit dem Ziel, Zeichenkette $\boldsymbol{x}$ in die Zeichenkette $\boldsymbol{y}$ zu transformieren. Der Abstand der beiden Ketten ist durch die Höhe der Kosten einer günstigsten Ausrichtung definiert. Der Smith-Waterman-Abstand bedient sich prinzipiell der gleichen Operationen, berechnet aber *lokale Ausrichtungen*. Es werden die ähnlichsten Teilabschnitte der Eingabesequenzen gesucht, ohne den Anspruch zu erheben, die vollständigen Sequenzen ineinander transformieren zu müssen. Diese Eigenschaften sind günstig für den Vergleich kürzester Dependenzpfade, da die Pfade nicht gleich lang und stellenweise recht unterschiedlich sein können, ohne unmittelbar als völlig unähnlich ange-

sehen zu werden. Das ermöglicht den positiven Einfluss der wichtigen Teile im Pfad. Dies ist eine Eigenschaft, die etwa dem ursprünglichen Kürzester-Pfad-Ansatz von Bunescu und Mooney (2005) fehlt: Der dort vorgestellte Kernoperator definiert den Kernwert zweier ungleich langer Pfade bzw. Sequenzen strikt als Null. Im Gegensatz zur Levenshtein-Metrik, die Abstände misst, stellt das Smith-Waterman-Maß außerdem ein Ähnlichkeitsmaß dar, nimmt also große Werte bei Ähnlichkeit an.

Das Smith-Waterman-Maß definiert die Ähnlichkeit $\mathrm{SW}(\boldsymbol{x}, \boldsymbol{y})$ zweier Zeichenketten als den Wert der höchsten lokalen Übereinstimmung $\mathrm{s}(\boldsymbol{x}, \boldsymbol{y}, \pi)$ der Zeichenketten $\boldsymbol{x}$ und $\boldsymbol{y}$:

$$\mathrm{SW}(\boldsymbol{x}, \boldsymbol{y}) \stackrel{def}{=} \max_{\pi \in \mathcal{A}(\boldsymbol{x}, \boldsymbol{y}, \pi)} \mathrm{s}(\boldsymbol{x}, \boldsymbol{y}, \pi),$$

wobei π eine lokale Ausrichtung aus der Menge $\mathcal{A}(\boldsymbol{x}, \boldsymbol{y}, \pi)$ aller lokaler Ausrichtungen zwischen den Zeichenketten $\boldsymbol{x}$ und $\boldsymbol{y}$ bezeichne. Analog zu Maßen wie dem Levenshtein-Abstand existiert für die Berechnung des Smith-Waterman-Maßes ein Algorithmus des dynamischen Programmierens. Er ist gegeben durch

$$\mathrm{SW}(i, j) = \max \begin{cases} 0 \\ \mathrm{SW}(i-1, j-1) + d(\boldsymbol{x}_i, \boldsymbol{y}_j) \\ \mathrm{SW}(i-1, j) - G \\ \mathrm{SW}(i, j-1) - G, \end{cases} \tag{4.8.1}$$

wobei $d(\boldsymbol{x}_i, \boldsymbol{y}_j)$ die Kosten für einen Austausch der Zeichen $\boldsymbol{x}_i$ und $\boldsymbol{y}_i$ und G die Strafe einer Auslassung bzw. Einfügung bezeichnet. Dies entspricht der Originaldefinition des Maßes von Smith und Waterman (1981). Es ist allerdings auch möglich, verschiedene Strafgrößen für Auslassung und Löschung einzuführen, was im Zuge der Kernoperatordefinition auch geschehen wird.

Um aus dem Smith-Waterman-Maß einen gültigen Kernoperator zu gewinnen, schlagen Saigo u. a. (2004) vor, alle lokalen Ausrichtungen wie folgt aufzusummieren:

$$K_{\mathrm{LA}}(\boldsymbol{x}, \boldsymbol{y}) \stackrel{def}{=} \sum_{\pi \in \mathcal{A}(\boldsymbol{x}, \boldsymbol{y}, \pi)} e^{\beta \cdot \mathrm{s}(\boldsymbol{x}, \boldsymbol{y}, \pi)}, \quad \beta \geq 0. \tag{4.8.2}$$

Es stellt sich nun die Frage der Parameterwahl für das Smith-Waterman-Maß, wobei insbesondere die Werte $d(\boldsymbol{x}_i, \boldsymbol{y}_j)$ wichtig sind. Diese Werte entstammen einer Matrix der Dimension $|\mathcal{V}| \times |\mathcal{V}|$ von Austauschparametern, wenn $\mathcal{V}$ ein Vokabular von Zeichen darstellt. Katrenko und Adriaans (2008) schlagen zur Definition der *Austauschmatrix* $d(\cdot, \cdot)$ die Verwendung von wahrscheinlichkeitsbasierten Kookkurrenzmaßen vor,

wie sie bereits in Abschnitt 4.7 definiert und verwendet wurden. Dazu werden analog zum eben genannten Abschnitt Wahrscheinlichkeitswerte für das Bestehen einer Kookkurrenzrelation zwischen je zwei Tokens des Vokabulars $\mathcal{V}$ berechnet. Um einen Wert für die Austauschbarkeit zweier Tokens t_1 und t_2 zu erhalten, werden ihre Verteilungen t_1^r und t_2^r mittels eines Verteilungsähnlichkeitsmaßes berechnet. Die von Katrenko und Adriaans (2008) gewählten Maße sind in Tabelle 4.4 aufgeführt. Zusätzlich zu den Maßen Cosinus- sowie Dice-Ähnlichkeit wird im Aufsatz auch das L2-Maß aufgelistet.

Es ist dabei wichtig zu beachten, dass das L2-Maß ein Abstandsmaß und kein Ähnlichkeitsmaß darstellt. Um einen Ähnlichkeitswert aus dem L2-Abstand zu erhalten, wird dieser auf Einheitssumme normiert und von Eins subtrahiert. Ähnlich kann man mit den Abstandsmaßen aus Tabelle 4.3 verfahren, die im Kontext des Verteilungskerns verwendet werden. Die Jensen-Shannon-Divergenz stellt dabei einen Spezialfall dar, da sie Werte in $[0, 2]$ annimmt, wenn sie wie in der Tabelle abgebildet formuliert ist. Daher muss der Divergenzwert zunächst durch Zwei geteilt und dann von Eins subtrahiert werden, um einen Ähnlichkeitswert in $[0, 1]$ zu erhalten.

Um diese Mittel zur Berechnung von Ähnlichkeitswerten zwischen Dependenzpfaden verwenden zu können, muss die Funktion d allerdings bezüglich der Dependenzkanten angepasst werden. Die Pfade $\boldsymbol{x}$ und $\boldsymbol{y}$ seien als Folge von Zeichenketten gegeben: $\boldsymbol{x} = x_1 x_2 \dots x_n$ und $\boldsymbol{y} = y_1 y_2 \dots y_m$, wobei jeder Pfad mit einem Text-Token beginnt sowie aufhört und zwischen je zwei Tokens aus $\mathcal{V}$ in einem Pfad eine Dependenzkantenbezeichnung steht, die nicht im Kookkurrenzvokabular $\mathcal{V}$ enthalten ist. Die Austauschfunktion d' sei dann definiert als

$$d'(\boldsymbol{x}_i, \boldsymbol{y}_j) = \begin{cases} d(\boldsymbol{x}_i, \boldsymbol{y}_j) & \boldsymbol{x}_i, \boldsymbol{y}_j \in \mathcal{V} \\ 1 & \boldsymbol{x}_i, \boldsymbol{y}_j \notin \mathcal{V}, \quad \boldsymbol{x}_i = \boldsymbol{y}_j \\ 0 & \boldsymbol{x}_i, \boldsymbol{y}_j \notin \mathcal{V}, \quad \boldsymbol{x}_i \neq \boldsymbol{y}_j \\ 0 & \boldsymbol{x}_i \in \mathcal{V}, \quad \boldsymbol{y}_j \notin \mathcal{V} \\ 0 & \boldsymbol{x}_i \notin \mathcal{V}, \quad \boldsymbol{y}_j \in \mathcal{V} \end{cases}$$

und wird anstatt der Ursprungsfunktion d für die Berechnung des Smith-Waterman-Maßes in Gleichung 4.8.1 verwendet. Zudem kann, wie schon erwähnt, der Strafterm G in zwei Einzelstrafen G_1 und G_2 für Löschung bzw. Einfügung aufgeteilt werden. Für die Berechnung des Kernwertes zweier Pfade potentieller Relationen $\boldsymbol{x}$ und $\boldsymbol{y}$ werden die lokalen Ausrichtungen zwischen beiden Sequenzen schließlich gemäß Gleichung 4.8.2 summiert.

Maß	Definition
Cosinus	$d(t_i, t_j) = \frac{\langle t_i^r, t_j^r \rangle}{\|t_i^r\| \cdot \|t_j^r\|} = \frac{\sum_{c \in \mathcal{C}} P(c\|t_i) P(c\|t_j)}{\sqrt{\sum_{c \in \mathcal{C}} P(c\|t_i)^2 \sum_{c \in \mathcal{C}} P(c\|t_j)^2}}$
Dice	$d(t_i, t_j) = \frac{2\|F(t_i) \cap F(t_j)\|}{\|F(t_i)\| + \|F(t_j)\|}, \qquad F(t_i) \stackrel{def}{=} \{ c \mid c \in \mathcal{C}, P(c\|t_i) > 0 \}$

Tabelle 4.4: Cosinus- und Dice-Verteilungsmaße, die zur Erstellung der Austauschmatrix $d(\cdot, \cdot)$ verwendet werden. Das Cosinusmaß ist direkt durch den Cosinuswert des Winkels zwischen zwei Verteilungsvektoren gegeben.

4.9 Graph-Kern

(Airola u. a. 2008)

Die bisher vorgestellten Techniken zur Merkmalgewinnung und Kernoperatordefinition berücksichtigen nur textuell oberflächliche Informationen oder kleinere Teile des Syntaxbaums. Der Verteilungskern (Abschnitt 4.7) macht sich etwa nur Informationen bezüglich potentieller Relationsargumente zu Nutze, die Walkmerkmale (Abschnitt 4.5) und der Local-Alignment-Kern (Abschnitt 4.8) beschränken sich auf den kürzesten Pfad zwischen zwei Entitäten. Airola u. a. (2008) stellen fest, dass aber auch der kürzeste Dependenzpfad zwischen zwei Entitäten wichtige Informationen auslassen kann, die im Dependenzgraphen, dem der Pfad entnommen ist, jedoch vorhanden sind. Ein Ausweg wäre an dieser Stelle zunächst ein Tree-Kernel, der die strukturelle Ähnlichkeit zweier Bäume berechnen kann. Das schließt allerdings die Verwendung von Parsingmethoden aus, die Kreise generieren können, wie sie durch die „collapsed“ Darstellung des Stanford-Schemas generiert werden können (Marneffe u. a. 2006). Es ist deshalb von Interesse, einen Kernoperator zu finden, der auf *vollständigen, allgemeinen Graphen* definiert ist. Ein solcher Kern wird in diesem Abschnitt vorgestellt.

Die Shortest Path Hypothesis geht davon aus, dass die für die Relationsextraktion wichtigsten Informationen auf dem kürzesten Pfad zwischen den Argumenten der Relation konzentriert seien. Bunescu und Mooney (2005) weisen insbesondere darauf hin, dass das wichtige Prädikat auf diesem Pfad läge, selbst bei komplizierten Dependenzstrukturen. Auch wenn diese Behauptung in vielen Fällen korrekt sein mag, so lässt sich aber auch anhand sogar recht einfacher Beispiele zeigen, dass es Ausnahmen gibt, wie in Abbildung 4.3 zu sehen ist. Die kürzesten Pfade dort enthalten teilweise gar kein Prädikat sondern stellen nur eine minimale Dependenzverbindung zwischen den beiden Entitäten dar. Im Beispiel oben rechts ist das Prädikat zwar enthalten, liefert aber eine verwirrende Information, da es isoliert betrachtet wird. Die Information

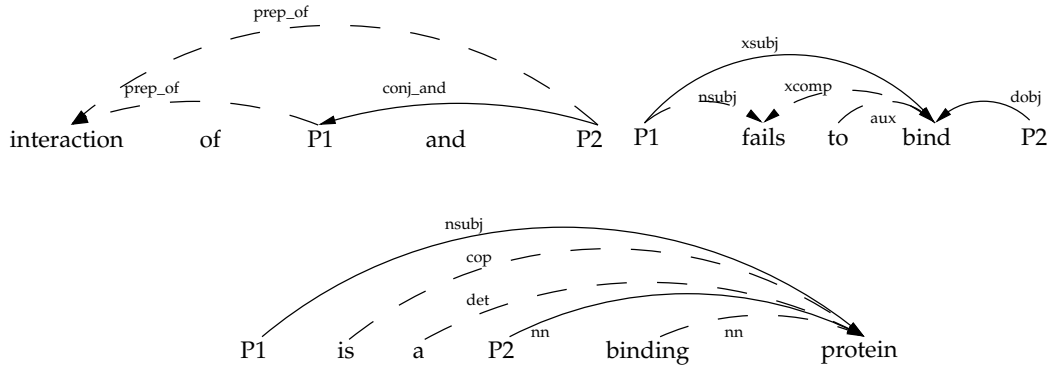

Abbildung 4.3: Teile von Sätzen, die mit dem Stanfordparser analysiert wurden. Durch spezielle Abhängigkeitsstrukturen kann es vorkommen, dass der kürzeste (ungerichtete) Pfad, der hier mit durchgezogenen Kanten markiert ist, zwischen zwei Entitäten wichtige Informationen auslassen kann. In den Beispielen oben links und unten verfügt der kürzeste Pfad über keine lexikalisch-semantische Information darüber, in welcher Beziehung die jeweiligen Argumente zueinander stehen. Im Teilsatz, der oben rechts abgebildet ist, ist sogar insofern eine falsche Information enthalten, als dass der negierende Ausdruck '*fails to*' fehlt, während das Prädikat '*bind*' isoliert im Pfad enthalten ist.

'*fails to*', die zur Negation der Bedeutung des Prädikats dient, ist im Pfad exkludiert und leistet deshalb in einem Verfahren, dass nur auf kürzesten Pfaden arbeitet, keinen Beitrag.

Wie bereits in Abschnitt 4.8 kurz angedeutet wurde, sind wesentliche Probleme im Vergleich von Dependenzparses ihre unterschiedliche Größe sowie Struktur. Der Local Alignment Kern löst das Problem der verschiedenen Pfadlängen durch den Einsatz eines Maßes, das mit Löschungen und Einfügungen arbeiten kann. Dabei handelt es sich allerdings um ein Sequenzmaß, das für den Einsatz mit allgemeinen Graphen nicht ohne Weiteres funktioniert. Airola u. a. (2008) verwenden als Grundlage ihres Verfahrens Teile einer bereits existierenden, mathematischen Methode, die auf Graphen in Form einer *Adjazenzmatrix* den eigentlichen Kernwert berechnet. Der erste wesentliche Schritt ist daher, den Dependenzgraphen eines Satzes in eine solche Darstellung zu überführen. Von zentraler Bedeutung ist die letztendliche Vergleichbarkeit der Graphmatrizen, die deshalb idealerweise die gleichen Dimensionen besitzen.

Airola u. a. (2008) konvertieren den Dependenzgraphen nicht direkt in die Adjazenzmatrixdarstellung. Um möglichst viel Information zu kodieren, wird ein gewichteter Graph erstellt, der aus mindestens *zwei Zusammenhangskomponenten* besteht. Die erste Zusammenhangskomponente soll dabei die Graphstruktur selbst wiedergeben, die zweite dient zur Kodierung des Satzes als Sequenz von Tokens. Ein Beispiel für die

folgende Beschreibung der Graphkonstruktion ist in Abbildung 4.4 gegeben.

Der **erste Subgraph** wird auf Basis des Dependenzgraphen erstellt. Jeder Knoten im Zielgraphen wird durch eine Menge von Markierungen bzw. Labels[8] identifiziert. Für jeden Knoten im Dependenzgraphen gibt es einen korrespondierenden Knoten in der ersten Zusammenhangskomponente. Der Zielknoten erhält als Labels die Tokenbezeichnung des Ursprungsknotens sowie seine Wortartklasse. Eine Ausnahme von dieser Regel bilden Gen- und Proteinnamen, die für die bessere Verallgemeinerung des Verfahrens maskiert werden. Der erste Argumentkandidat erhält das Label '*PROT1*', das zweite '*PROT2*'. Alle anderen Protein- oder Gennamen, die aktuell nicht als potentielle Argumente aufgefasst werden, werden durch das Label '*PROT*' dargestellt. Befindet sich das Token auf dem kürzesten Pfad zwischen den aktuell betrachteten Relationsargumenten, erhalten beide Labels den Zusatz '*IP*' (In Path). Des Weiteren wird für jede Kante im Dependenzgraphen ein neuer Knoten im ersten Subgraphen eingeführt. Dieser Knoten erhält als Label die Beschriftung der Dependenzkante. Falls die Kante auf dem kürzesten Pfad liegt, wird ebenfalls der Zusatz 'IP' vergeben. Die Kanten im Zielgraphen werden so definiert, dass jeder Knoten, der von einer Kante im Dependenzgraphen abgeleitet wurde, eine eingehende Kante von dem Knoten erhält, der den Kopf der Dependenzrelation darstellt und eine ausgehende Kante zu dem Knoten besitzt, dessen Token der Dependent der Relation ist. Zudem erhalten die Kanten im Zielgraphen Gewichte, wobei sich das Gewichtungsschema nach der Hypothese des kürzesten Pfads richtet. Alle Kanten im Zielgraphen, die Elemente miteinander verbinden, welche im originalen Dependenzgraphen auf dem kürzesten Pfad liegen, erhalten ein Gewicht von 0.9. Alle anderen Kanten werden mit 0.3 gewichtet. Dieses Schema für die Gewichtsverteilung wurde in kleineren Experimenten ermittelt und folgt keiner tieferen mathematischen Motivation.

Der **zweite Subgraph** gibt die lineare Struktur des zugrunde liegenden Satzes wieder. Er besteht aus je einem Knoten für jeden Knoten aus dem ursprünglichen Dependenzgraphen. Die Knoten des Subgraphen erhalten als Labels wieder sowohl die Tokenbezeichnung selbst als auch die Wortart des Tokens. Protein- und Gennamen werden analog zur ersten Zusammenhangskomponente mit '*PROT1*', '*PROT2*' und '*PROT*' maskiert. Die Knoten erhalten einen Labelzusatz 'B'(efore), 'M'(iddle) oder 'A'(fter), der sich nach der Position des entsprechenden Tokens relativ zu den Argumenten der zu extrahierenden PPI richtet. Steht ein Token im Satz vor beiden Argumenten, so erhält es den Zusatz 'B', erscheint es zwischen den beiden Argumenten, wird ein 'M' hinzugefügt. Der Zusatz 'A' dient als Indikator dafür, dass das entsprechende Token erst

[8] Als Label wird an dieser Stelle eine Zeichenkette zu Identifikationszwecken bezeichnet und ist nicht mit einem Klassenlabel zu verwechseln.

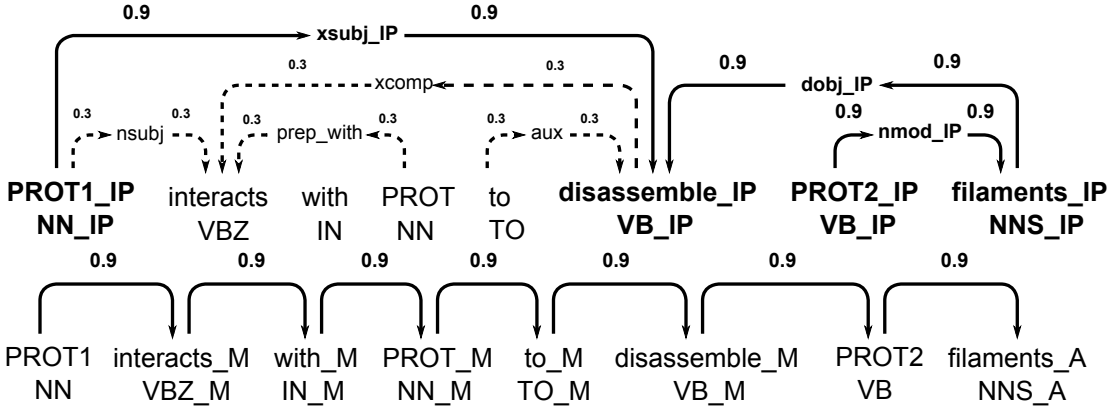

Abbildung 4.4: Graphrepräsentation, die auf Grundlage eines Beispielsatzes generiert wurde. Das Kandidatenpaar ist mit PROT1 bzw. PROT2 maskiert, der kürzeste Pfad ist mittels durchgezogener Linien dargestellt. Der Graph besteht aus zwei Zusammenhangskomponenten, wobei die erste die Dependenzgraphstruktur des Satzes repräsentiert und die zweite die Wortsequenz des Satzes kodiert. Die Knoten des ersten Subgraphen erhalten den Labelzusatz 'IP', falls sie auf dem kürzesten Pfad zwischen den Relationsargumenten liegen. Die Knoten des zweiten Graphen erhalten den Labelzusatz 'B'(efore), 'M'(iddle) oder 'A'(fter), je nachdem ob sie vor, zwischen oder nach den beiden betrachteten Relationsargumenten stehen.

nach beiden Argumenten im Satz genannt wird. Um die Reihenfolge der Knoten auszudrücken, bekommt der Knoten v_i je eine ausgehende Kante zum Knoten v_{i+1}, wobei sich die Indizes i und $i+1$ an der Wortreihenfolge im Originalsatz orientieren. Diese Kanten erhalten jeweils ein Gewicht von 0.9.

Für die mathematische Verarbeitung des so gewonnenen Graphen muss die Darstellung zu einer Adjazenzmatrix konvertiert werden. Der oben beschriebene Graph sei mit $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ bezeichnet. Die Knotenmenge $\mathcal{V}$ enthalte alle Knoten des Graphen, $\mathcal{E}$ bestehe aus der Menge aller Kanten. Zwei Knoten sind dabei nur dann gleich, wenn sie vom gleichen Element des Dependenzgraphen abgeleitet wurden und sich in der gleichen Zusammenhangskomponente befinden. Haben zwei Knoten in $\mathcal{G}$ die gleichen Labels, folgt nicht zwingend ihre Gleichheit. Zudem enthalte die Menge $\mathcal{L} = \{ l_1, \ldots, l_r \}$ alle möglichen Labels, die an Knoten des konstruierten Graphen vergeben werden können.

Die Adjazenzmatrix $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ wird dann durch die Indexmenge $\{ 1, \ldots, |\mathcal{V}| \}$ indiziert, wobei der Index i dem Knoten $v_i \in \mathcal{V}$ entspreche. Die Elemente a_{ij} der Adjazenzmatrix sind durch das Kantengewicht der Kante von $v_i \in \mathcal{V}$ nach $v_j \in \mathcal{V}$ gegeben. Falls keine Kante zwischen v_i und v_j existiert, so gilt $a_{ij} = 0$. Es wird zudem eine Labelzuweisungsmatrix $L \in \mathbb{R}^{|\mathcal{L}| \times |\mathcal{V}|}$ eingeführt. Diese Matrix ist eine Indikatormatrix, d.h. sie enthält ausschließlich Einsen und Nullen. Sie zeigt für einen Knoten v_i an, ob

er das Label l_j besitzt ($[L]_{ij} = 1$) oder nicht ($[L]_{ij} = 0$). Auf Grund der hohen Anzahl von möglichen Labels gegenüber der geringen Anzahl von Labels, die konkret an einen einzelnen Knoten vergeben werden, ist diese Matrix extrem dünn besetzt.

Man betrachte nun die quadrierte Adjazenzmatrix A^2. Aus der Graphentheorie ist bekannt, dass $[A^2]_{ij}$ die Summe der multiplizierten Kantengewichte aller Pfade von Knoten v_i zu v_j darstellt, wobei jeder Pfad über einen einzelnen Mittelknoten v_k führt und damit eine Länge von zwei Kanten besitzt. Dies lässt sich verallgemeinern, so dass die Matrix A^n die Summe der multiplizierten Gewichte aller Kanten aus Pfaden der Länge n zwischen zwei Knoten aus $\mathcal{V}$ beinhaltet. Durch diese Methode lassen sich allgemein Strukturinformationen eines Graphen darstellen. Allerdings gibt es kein spezielles Interesse an Pfaden einer bestimmten Länge. Das Ziel ist eine Matrix, die die Gewichte aller möglichen Pfade von Knoten v_i zu Knoten v_j aufsummiert darstellt.

Um das zu erreichen, wird die Neumann-Reihe einer Matrix betrachtet. Sie ist definiert als

$$(E - A)^{-1} = E + A + A^2 + \cdots = \sum_{k=0}^{\infty} A^k,$$

falls $|A| < 1$, wobei $|A|$ den Spektralradius von A bezeichne. In der obigen Gleichung steht E außerdem für die Einheitsmatrix. Um in dieser Darstellung Schleifen zu eliminieren, die einen Knoten mit sich selbst verbinden, wird die Einheitsmatrix subtrahiert:

$$W = (E - A)^{-1} - E.$$

Die Matrix W beinhaltet nun die Summe aller Pfade, die zwei Knoten v_i und v_j in der ursprünglichen Adjazenzmatrix A miteinander verbinden.

Es stellt sich an dieser Stelle die Schwierigkeit dar, dass die Matrizen W zweier potentieller PPI nicht die gleiche Dimension besitzen, falls ihre Sätze unterschiedlich viele Tokens beinhalten. Ein Weg, um die Vergleichbarkeit herzustellen, besteht in der Multiplikation von W mit der Labelzuweisungsmatrix L auf folgende Weise:

$$G = LWL^\top. \tag{4.9.1}$$

Die resultierende Matrix G ist von der Dimension $\mathcal{L} \times \mathcal{L}$. Dies ist unabhängig davon, welche Dimension die Matrix W hatte, so dass alle so transformierten Matrizen die gleichen Ausmaße besitzen. Um einen Ähnlichkeitswert zweier Matrizen G^1 und G^2 zu berechnen, werden ihre Elementprodukte aufsummiert:

$$K(G_1, G_2) \stackrel{def}{=} \sum_{i=1}^{|\mathcal{L}|} \sum_{j=1}^{|\mathcal{L}|} [G^1]_{ij} \cdot [G^2_{ij}].$$

Der so gewonnene Kernwert kann als die Summe der Stärke von Labelkombinationen je zweier Knoten aus beiden Graphen interpretiert werden. Mit Stärke ist dabei die Stärke ihrer Verbindung gemeint, die durch die Kantengewichte gegeben ist, die zwei Knoten miteinander verbinden, die über diese Labels verfügen. Solche Verbindungen gibt es in einem Graphen potentiell mehrere, da ein einzelnes Label natürlich mehrfach vergeben werden kann.

KAPITEL 5

Methodische Grundlagen der Evaluation

In diesem Kapitel werden Methoden und Problempunkte der Systemauswertung im Rahmen der PPI-Extraktion besprochen. In Abschnitt 5.1 werden einige wichtige Punkte betreffend die prinzipielle Methodik der Auswertung von Systemen besprochen, die sich der PPI-Extraktion widmen. Abschnitt 5.2 beschäftigt sich anschließend mit der Wahl von freien Parametern, die sowohl im Lernalgorithmus für SVMs als auch in den unterschiedlichen Techniken enthalten sind, die zusammen mit der SVM verwendet werden. Abschnitt 5.3 erläutert ein Nachprozessierungsverfahren, das die Handhabung ungleicher Datenverteilungen erleichtert.

5.1 Evaluation

Die Entwicklung neuer Verfahren macht in praktischen Anwendungen nur Sinn, wenn es ein *Maß für die Ergebnisgüte* der Verfahren gibt. Im Bereich des maschinellen Lernens und insbesondere der Informationsextraktion gibt es traditionelle Maße, um die Leistung eines Klassifikators zu beurteilen. Da Datenmaterial zum Anlernen und Testen eines Klassifikators allgemein rar und teuer ist, wurden früh Strategien entwickelt, um vorhandenes Material optimal auszunutzen. Diese Grundmethodik soll hier beschrieben werden. Darüber hinaus gibt es anwendungsspezifische Probleme der Evaluation, was die Vergleichbarkeit von Ergebnissen unterschiedlicher Entwickler betrifft. Diese Schwierigkeiten sollen kurz angesprochen werden, um letzten Endes die Art und Weise der Evaluation in dieser Arbeit zu motivieren.

Um die Leistung eines Klassifikators zu beurteilen, lässt man ihn gewöhnlicherweise einen Satz von Testdaten bearbeiten. Für diese Testdaten sind die richtigen Antworten bereits bekannt, so dass nach erfolgter Klassifikation ein Vergleich zwischen

Klassifikatorausgabe und korrekten Antworten stattfinden kann. Im Falle der PPI werden dabei die klassischen Maße des Information Retrieval verwendet, da es unter anderem darum geht, relevante Objekte von nicht relevanten Objekten zu trennen. Die dabei verwendeten Maße heißen *Precision (Genauigkeit)*, *Recall (Ausschöpfung)* und *F-Maß (harmonisches Mittel aus Precision und Recall)*. Um einen Wert für diese Maße zu erhalten, wird auf die Klassifikationsstatistik zurückgegriffen. Von Interesse ist, wie viele Objekte aus dem Problemkreis $\mathbf{\Omega}$ korrekt klassifiziert wurden (*True Positive (TP)*), wie viele Objekte der falschen Klasse zugewiesen wurden (*False Positive (FP)*) und wie viele Objekte nicht erkannt wurden, obwohl sie einer Klasse von Interesse angehören (*False Negative (FN)*). Technisch gesehen ist der letzte Fall ebenfalls eine falsche Klassenentscheidung. Wie in Abschnitt 3.2 erklärt wurde, wird eine Klasse aber anwendungsabhängig auch als Rückweisungsklasse verstanden, was im Fall der PPI-Extraktion der Nicht-Relevanz des rückgewiesenen Objekts entspricht. Im Fall der Relationsextraktion bedeutet die Zugehörigkeit eines Objekts zu dieser Klasse, dass der Relationskandidat, dem Klassifikator nach zu urteilen, keine Relation darstellt. Die Maße Recall, Precision und F-Maß sind dann wie folgt definiert:

$$P(recision) = \frac{TP}{TP + FP} \qquad R(ecall) = \frac{TP}{TP + FN} \qquad F(\text{-}Maß) = \frac{2PR}{P + R}.$$

Die naheliegendste Auswertung eines Lernalgorithmus' ergibt sich durch Aufteilung des gesamten verfügbaren Materials in zwei Teile. Mit verfügbarem Material sind dabei Daten gemeint, für die die richtigen Antworten bzw. Klassenzugehörigkeiten bekannt sind. Die beiden Datenpartitionen sind dabei normalerweise nicht von gleicher Größe. In der Regel gilt, dass ein statistischer Klassifikator besser wird, je mehr Beispielmaterial zur Verfügung steht. Typischerweise partitioniert man deshalb in einen Teil mit ca. 90% der Daten zum Lernen und einen Teil von ca. 10% für den Test.

Um das vorhandene Datenmaterial jedoch besser auszunutzen, wird in der Praxis meistens eine *Kreuzvalidierung* vorgenommen. Für eine n-fache Kreuzvalidierung wird das vorhandene Datenmaterial in n gleich große, überlappungsfreie Teile partitioniert. In der Praxis trifft man oft auf 10-fache Kreuzvalidierungen als Kompromiss zwischen Datenausnutzung und Laufzeit des Auswertungssystems. Anschließend werden je $n - 1$ Partitionen für das Training verwendet, die überbleibende Partition dient zum Test. Dieses Verfahren wird wiederholt, bis jede Partition einmal als Testmaterial fungiert hat. In jeder der n Runden werden die Maße P, R und F berechnet. Für das Ergebnis der Kreuzvalidierung werden die Statistiken TP, FP und FN jeder Runde akkumuliert. Das Resultat ergibt sich schließlich aus der Berechnung der oben genannten Maße auf den so gewonnenen Summen.

Ein Spezialfall der Kreuzvalidierung ist die *Leave-One-Out-Validierung*. Damit wird

eine Kreuzvalidierung mit maximaler Partitionenanzahl bezeichnet. Im Extremfall dient also in jeder Runde genau ein Objekt als Testdatum. Die Kreuzvalidierung kann allerdings je nach Anwendungsfall auf größeren Gesamtstrukturen definiert werden, etwa auf Satz- oder Dokumentenebene, so dass die Partitionen potentiell mehrere Objekte - beispielsweise mögliche Relationen - umfassen.

Obwohl diese Maße und Verfahren allgemein anerkannt und häufig genutzt werden, gibt es Schwierigkeiten hinsichtlich der Vergleichbarkeit der Ergebnisse. Dadurch, dass im Falle der PPI-Extraktion viele Vorverarbeitungsschritte nötig sind (vgl. Abschnitt 3.1), unterscheiden sich meistens bereits die zugrunde liegenden Daten zweier Systeme. Weniger signifikante Unterschiede sind bereits auf der Ebene von Tokens zu suchen, falls die originale Tokenisierung des Materials nachbearbeitet oder vollständig neu erstellt wurde. Sehr große Abweichungen gibt es auf der Satzstrukturebene durch den Einsatz unterschiedlicher Parser. Da viele Verfahren auf den Strukturbäumen bzw. -graphen von Sätzen arbeiten, hat die Gestalt der Graphen einen großen Einfluss auf die Ergebnisse eines Klassifikators. Solche Unterschiede bezüglich der Vorprozessierung sind kaum zu vermeiden, da nicht alle Werkzeuge, wie etwa Parser, immer vorhanden sind. Für den Vergleich eines Systems mit einem anderen ist es deshalb sinnvoll - soweit möglich - das gleiche, vollständig vorprozessierte Material zu verwenden.

Im Rahmen der PPI-Extraktion fällt dem NE-Tagger eine besondere Bedeutung zu. Auf Grundlage eines gegebenen Textes werden die Relationskandidaten in diesem Text typischerweise durch die Kombination aller Gen- oder Proteinnamen innerhalb eines Satzes gebildet. Variiert die Anzahl der Entitäten in einem Text, etwa durch unterschiedliche Modelle oder Fehlerkennung der NE-Tagger, so variiert auch die Anzahl von potentiellen Relationen. Auch wenn die fehlenden Relationskandidaten negative Relationen, im Falle von PPI also keine Interaktion darstellen, so wirkt sich die geringere Anzahl dennoch potentiell auf das Ergebnis eines Systems aus. Das ist dann der Fall, wenn ein System eine Nicht-Interaktion fälschlicherweise als Interaktion erkennt und somit ein False Positive (FP) produzieren würde. Zudem muss entschieden werden, welche Art von Relationen zugelassen sind. In dieser Arbeit wird stets von zweistelligen Relationen ausgegangen. Allgemein sind aber auch einstellige Relationen denkbar, die im Bereich der PPI beispielsweise Selbstinteraktionen bezeichnen. All diese Faktoren wirken sich auf die Anzahl der Relationskandidaten in einem gegebenen Text aus und damit auf die Anzahl der Objekte, die einem Klassifikator zur Evaluation übergeben werden. Beim Vergleich zweier Extraktionssysteme müssen diese Fragen der Interpretation der Daten berücksichtigt werden, soweit das überhaupt möglich ist.

Einen weiteren wichtigen Punkt im Zusammenhang mit der Evaluation stellt die Aufteilung der Daten in Partitionen für die Kreuzvalidierung dar. In einem Satz mit n

Entitäten gibt es $\binom{n}{2}$ Paare von Entitäten. Jedes Paar stellt eine potentielle Relation und damit ein Objekt für eine Klassenzuweisungsentscheidung dar. Soll nun eine Kreuzvalidierung durchgeführt werden, besteht der erste Ansatz darin, eine große Menge aller Relationskandidaten jedes Satzes im Datenmaterial zu bilden. Auf dieser Menge werden die Partitionen angelegt und anschließend wird die Evaluation durchgeführt. Im Falle der PPI-Extraktion mit Methoden des maschinellen Lernens führt diese Vorgehensweise allerdings zu einer zu optimistischen Schätzung der Systemleistung. Um das nachzuvollziehen betrachte man folgenden Satz, der bereits in Kapitel 2 aufgeführt wurde:

> ' The ability of *c-Fos* and *c-Jun* proteins to interact directly with the *TATA box-binding protein* (*TBP*), the general transcription factor required for initiating the assembly of transcription complexes, was investigated. '

Wie bereits zuvor beschrieben, existieren an dieser Stelle vier PPI. Diese Interaktionen finden zwischen *c-Fos* und *TATA box-binding protein* sowie *TBP* und zwischen *c-Jun* mit den gleichen Interaktionspartnern statt. Bei oben beschriebener Aufteilung auf Objektebene der Daten kann es vorkommen, dass einige dieser Relationen dem Trainingsmaterial und die restlichen dem Testmaterial zugeordnet werden. Da die Kandidaten dem gleichen Satz entstammen, sind die Grundlageninformationen nahezu die gleichen, sogar die Dependenzpfade ähneln sich in hohem Maße. Eine Methode wie der Graph-Kernel weist diesen Relationskandidaten unmittelbar hohe Ähnlichkeitswerte zu, da die Graphstruktur der Syntaxanalyse stets die gleiche ist. Damit bekommt ein Klassifikator, der auf einer solchen Partitionierung ausgewertet wird, unzulässige Hinweise, die seine Ergebnisse verbessern. Unzulässig sind diese Hinweise deshalb, weil es in einer realistischen Anwendung nicht geschehen wird, dass zu klassifizierende Daten teilweise aus den gleichen Sätzen stammen, die für das Lernen verwendet wurden.

Man sollte deshalb dafür Sorge tragen, dass zumindest auf Satzebene evaluiert wird, also alle Relationskandidaten eines Satzes entweder im Trainings- oder Testmaterial enthalten sind. Airola u. a. (2008) weisen aber darauf hin, dass selbst eine solche Partitionierung noch zu optimistische Leistungsschätzungen hervorbringen kann. Innerhalb eines Dokuments wird oft von den gleichen Entitäten gesprochen, die zumeist bezüglich der gleichen Relationen zueinander genannt werden, während andere Entitätenpaare niemals als interagierend beschrieben werden. Auch hier greift die obige Argumentation, dass dies unzulässige Hinweise an den Klassifikator bezüglich einer Realanwendung liefert. Deshalb wird die Evaluation in dieser Arbeit stets auf Dokumentenebene statt finden. In einem Leave-One-Out-Lauf würde also in jeder Evaluationsrunde auf genau einem Dokument mit allen darin enthaltenen Relationskandidaten

getestet.

Zuletzt soll darauf hingewiesen werden, dass ein und dasselbe Klassifikationssystem mit dem gleichen Datensatz unterschiedliche Ergebnisse hervorbringen kann, wenn die Kreuzvalidierungspartitionen variieren. Damit ist gemeint, dass verschiedene Objekte, die in einer ersten Partitionierung zu genau einer Partition gehörten, in einer zweiten Partitionierung auf mehrere Partitionen verteilt sein können. Solche Effekte werden typischerweise explizit in einer Anwendung eliminiert, treten aber zwischen zwei unterschiedlichen Anwendungen regelmäßig auf. Dadurch, dass sich das Trainings- und Testmaterial in zwei unterschiedlichen Partitionierungen in jeder Runde voneinander unterscheidet, variieren die Einzelergebnisse und folglich die Gesamtstatistik. Im Extremfall eines direkten Systemvergleichs müssen deshalb sogar die Partitionierungen aufeinander abgestimmt werden.

Durch die in diesem Abschnitt aufgestellen, einheitliche Auswertungsrichtlinien werden die oben genannten Schwierigkeiten umgangen. Es ist daher wichtig anzumerken, dass diese Arbeit den aussagekräftigen Vergleich der in Kapitel 4 vorgestellten Methoden ermöglicht.

5.2 Parameterselektion für SVM und Kernoperatoren

In dieser Arbeit wird Klassifikation mittels Kern-SVMs vorgenommen. Der Lernalgorithmus für SVMs mit Schlupfvariablen lässt die Wahl eines Parameters *'C'* offen (vgl. Gleichung 3.3.12). Er dient dazu, zwischen großer Verallgemeinerung, im Sinne eines großen Randes der Hyperebene, und minimalem Trainingsfehler, im Sinne von Objekten, die schon im Training auf der „falschen“ Seite liegen, abzuwägen. Er wird als Strafparameter interpretiert: Je höher der Wert C ist, desto weniger wird Fehlklassifikation toleriert.

Die verschiedenen Kernoperatoren haben ihrerseits Parameter. Der RBF-Kern (vgl. Gleichung 3.3.13) führt den Parameter γ ein, der sehr großen Einfluss auf das Erscheinungsbild der Trennfunktion nimmt, der Local-Alignment-Kern besitzt Strafparameter für Einfügung und Löschung sowie den Parameter β, der für die Summierung aller Alignments verwendet wird (vgl. Gleichung 4.8.2).

Dies sind Beispiele für freie Parameter, die in der Arbeit mit der SVM und vielen der Kernoperatoren zu belegen sind. In den meisten Fällen, insbesondere in den oben genannten, gibt es weder eine natürlich erscheinende „gute“ Belegung der Parameter noch eine Zielfunktion, die bezüglich dieser Parameter zu optimieren wäre. Die Zielfunktion stellt unmittelbar die Leistung des Klassifikators dar. Da aber keine mathematische Funktion existiert, die den Zusammenhang zwischen Parametern und

Klassifikatorleistung in Verbindung setzen würde, gibt es an dieser Stelle keine analytisch oder numerisch zufriedenstellende Methode der Parameterwahl.

Hsu u. a. (2008) schlagen deshalb einen experimentellen Ansatz vor, der als *Gittersuche* (engl. *Grid Search*) bezeichnet wird. Die Achsen des Parameterraums werden in dieser Methode logarithmiert, so dass die Parameter als Exponenten angegeben werden. Es wird ein Bereich für jeden Parameter angegeben, in dem nach einer guten Belegung gesucht werden soll. Anschließend werden der Klassifikator - hier die SVM - und die dazugehörige Methoden - hier die Kernoperatoren - mit jeder so gebildeten Parameterkombination belegt und ausgewertet. Ist ein bester Wert ermittelt, schlagen Hsu u. a. (2008) eine Verfeinerung des Gitters vor. Das geschieht beispielsweise dadurch, dass keine ganzzahligen Exponenten mehr verlangt werden, sondern in $\frac{1}{4}$-Schritten voran gegangen wird.

Als Beispiel diene die Verwendung des RBF-Kerns, der über den freien Parameter γ verfügt. Zusammen mit dem Strafparameter der SVM ergeben sich Paare (C, γ), die untersucht werden sollen. Für C soll der Bereich $2^{-10}, 2^{-9}, \ldots, 2^{8}$ untersucht werden, für γ der Bereich $2^{-6}, 2^{4}$. Die Wahl des Bereichs stützt sich dabei bereits auf kleinere Erfahrungswerte, die im Umgang mit SVMs schnell zu erlangen sind. Es folgt eine Menge von zu untersuchenden Paaren $(2^{-10}, 2^{-6}), (2^{-10}, 2^{-5}), \ldots, (2^{8}, 2^{3}), (2^{8}, 2^{4})$ für die Parameter (C, γ).

Für die einzelnen Auswertungen wird im typischen Kontext des maschinellen Lernens ein gesonderter Teil des vorliegenden Datenmaterials betrachtet, der als *Entwicklungssatz* (engl. *Development Set*) bezeichnet wird. Er ist von Trainings- und Testmaterial getrennt und dient zur Schätzung der Methodenparameter. Liegt nur wenig Datenmaterial vor, wie es im Bereich der PPI-Extraktion der Fall ist, so wird zur besseren Nutzung des Materials auf die Kreuzvalidierung zurückgegriffen, die in Abschnitt 5.1 beschrieben wurde. In der Praxis wird die Parameterwahl oft direkt durch Kreuzvalidierungen auf dem Gesamtmaterial durchgeführt und auf die Aussparung eines Entwicklungssatzes wird verzichtet. So schlagen auch Hsu u. a. (2008) vor, das oben beschriebene Verfahren der Gittersuche unter Verwendung von Kreuzvalidierungen durchzuführen. Das heißt, dass für jeden Gitterpunkt - beispielsweise $(C = 2^{-4}, \gamma = 2^{3})$ - eine Kreuzvalidierung mit den Parametern $C = 2^{-4}$ und $\gamma = 2^{3}$ durchgeführt wird. Das Evaluationsresultat wird mit diesem Gitterpunkt verknüpft, so dass nach der Suche der beste Evaluationswert zusammen mit dem entsprechenden Punkt im Parameterraum geliefert werden kann. Je nach Anwendung muss entschieden werden, welches Auswertungsmaß für die Suche verwendet wird. In dieser Arbeit werden die Parameter bezüglich des besten F-Maß-Ergebnisses exploriert.

5.3 Schwellwertverfahren

Im Bereich der PPI-Extraktion wird typischerweise der beste F-Maß-Wert bei der Evaluation eines Klassifikators gesucht. Recall und Precision wiegen dabei gleich und können sich gegenseitig ausgleichen. Dabei kann es vorkommen, dass eines der beiden Maße einen sehr niedrigen Wert annimmt. In der Praxis begegnet man häufig dem Problem des niedrigen Recalls. Es wird folglich nur ein kleiner Teil der gesuchten Objekte als Gegenstand von Interesse identifiziert. Dabei ist die Precision signifikant höher als der Recall, die erkannten Objekte sind also öfter richtig klassifiziert als überhaupt Objekte gefunden werden. Ist der Unterschied zwischen Precision und Recall hoch, so kann es sich für den Wert des F-Maßes positiv auswirken, ein so genanntes Schwellwertverfahren durchzuführen. Im Kontext dieser Arbeit ist damit eine Erhöhung des Recalls gemeint, wobei das allgemeine Verfahren nicht auf diese Richtung beschränkt ist. Ein solches Schwellwertverfahren manifestiert sich als Nachprozessierung von bereits klassifizierten Objekten und ist kein Element des Klassifikationsmodells selbst. Wird ein Objekt $\boldsymbol{x}$ der Rückweisungsklasse zugewiesen, also nicht als interessantes Objekt erkannt, so wird geprüft, wie sicher diese Entscheidung war. Handelt es sich um eine knappe Entscheidung, so wird die Klasse gesucht, die nach der Rückweisungsklasse den höchsten Entscheidungswert für $\boldsymbol{x}$ vorweisen kann. Anschließend wird $\boldsymbol{x}$ dieser Klasse zugewiesen.

Diese Vorgehensweise hat zwei Auswirkungen. Zum Einen werden Objekte von Interesse, die nur knapp von den interessanten Klassen abgelehnt wurden, einer von eben diesen Klassen zugewiesen. Geht man vom Zwei-Klassen-Fall aus, bei dem es nur um Erkennung oder Nicht-Erkennung geht, wurde damit eine vollständig richtige Nachentscheidung getroffen. Dadurch erhöht sich der Recall des Klassifikators. Aber auch Objekte, die korrekterweise, wenn auch nur knapp, zurückgewiesen wurden, werden als gesuchte Objekte kategorisiert. Durch diese Nachentscheidungen sinkt das Precision-Maß. Aufgabe des hier beschriebenen Schwellwertverfahrens ist es, einen Schwellwert zu finden, der Recall und Precision im Sinne des F-Maßes ideal ausbalanciert.

Da die SVM immer nur den Zwei-Klassen-Fall lösen kann, wird hier das Vorgehen für zwei Klassen beschrieben. Für den Fall von K Klassen müssen mehrere SVM-Modelle erstellt werden. Für Details der Mehrklassenklassifikation sei auf Schölkopf und Smola (2001) verwiesen. Es sei die Rückweisungsklasse mit -1 und die Klasse der erkannten Objekte mit $+1$ bezeichnet. Der hier verwendete Algorithmus stellt zunächst fest, ob ein Objekt in die Rückweisungsklasse -1 klassifiziert wurde. Ist dies der Fall, so wird geprüft, wie knapp diese Entscheidung war. Eine knappe Entscheidung zeichnet

sich dadurch aus, dass der klassifizierte Vektor $\boldsymbol{x}$ nah an der Hyperebene der SVM liegt. Die Entfernung von $\boldsymbol{x}$ zur Ebene $\mathcal{H}$ beträgt $(\langle \boldsymbol{x}, \boldsymbol{w} \rangle + b) / \|\boldsymbol{w}\|$ mit dem Normalenvektor $\boldsymbol{w}$ und der Verschiebung b, wie bereits in Unterabschnitt 3.3.2 ausgeführt wurde. Unterschreitet die Entfernung zur Hyperebene einen Schwellwert $\mathcal{T} > 0$, so wird $\boldsymbol{x}$ nachträglich der Klasse $+1$ der Objekte von Interesse zugeordnet. Je größer der Wert $\mathcal{T}$ gewählt wird, desto mehr Objekte, die zurückgewiesen wurden, erhalten dadurch einen Klassenwechsel. Wird $\mathcal{T}$ hoch genug gesetzt, werden demnach alle Objekte im Datenmaterial in die Kategorie von Interesse eingeteilt, der Recall erreicht 100%. Der Schwellwert $\mathcal{T}$ ist ähnlich dem SVM-Strafparameter C Gegenstand der Optimierung und wird im Rahmen dieser Arbeit mit dem Verfahren der Gittersuche (vgl. Abschnitt 5.2) ermittelt.

KAPITEL 6

Daten

Dieses Kapitel beschreibt die Daten, die zur Durchführung von Experimenten und zur Evaluation der implementierten Methoden dienten. Die verwendeten PPI-Corpora sind Aimed (Bunescu u. a. 2005), das JULIE Lab Genereg Corpus (Buyko u. a. 2008) sowie das LLL05-Challenge-Corpus (Nédellec 2005).

Das **AIMed-Corpus** stellt zur Zeit das de facto Standardcorpus für die Auswertung von Systemen für die PPI-Extraktion dar. Es ist aus medizinischen Texten zusammengestellt, die der *Database of Interacting Proteins (DIP)*[1] entnommen wurden. Diese Datenbank enthält Texte, die sich mit PPI auseinander setzen. Von den dort enthaltenen Dokumenten wurden für das Corpus 200 ausgewählt und annotiert, wobei keine thematische Fokussierung bezüglich des Inhalts der ausgewählten Texte statt fand. Die Annotation geschah halbautomatisch mit Hilfe eines bereits existierenden Werkzeugs, wobei 1101 PPIs und 4141 Proteinnamen markiert wurden. Durch das Werkzeug gemachte Fehler wurden manuell korrigiert. Diese 200 Dokumente enthielten der Meinung der Autoren nach zu wenige negative Beispiele (Bunescu u. a. 2005). Deshalb wurden manuell weitere 30 Texte ausgewählt, die zwar Sätze mit mehr als einer Entität - und damit Relationskandidaten - enthalten, aber keine PPI beschreiben. In seiner aktuellen Zusammensetzung besteht das AIMed-Corpus aus 225 Dokumenten. Pyysalo u. a. (2008) haben unter anderem das AIMed-Corpus in ein einheitliches XML-Format überführt und die konvertierten Corpora öffentlich verfügbar gemacht[2]. Neben den Grundannotationen wie Sätzen und Tokens enthält die XML-Version NEs, Wortarten und einen Charniak-Lease-Dependenzparse (Charniak und Lease 2005). Das Extraktionssystem, das im Rahmen dieser Arbeit erstellt und benutzt wird, verwendet das AIMed-Corpus im XML-Format. Es enthält 1000 positive und 4834 negative Beispiele von PPI.

[1] Auf die Datenbank kann über `http://dip.doe-mbi.ucla.edu/` zugegriffen werden.

[2] Die Corpora können auf `http://mars.cs.utu.fi/PPICorpora/` heruntergeladen werden.

Das **Genereg-Corpus** entstand am JULIE LAB Jena[3]. Es enthält NE-, PPI- und Triggerannotationen. Ein Trigger ist dabei ein Wort, etwa ein bestimmtes Verb, das eine im Corpus annotierte Relation anzeigt. In den Verfahren für diese Arbeit wird allerdings kein Gebrauch von Triggerwörtern gemacht. Das Genereg-Corpus liegt für diese Arbeit in einer Version vor, die mit dem MST-Parser (Mcdonald u. a. 2006) syntaktisch analysiert wurde. Für die Erstellung des Corpus wurde die MEDLINE-Datenbank mittels einiger Suchanfragen konsultiert. Der Fokus der Anfragen lag dabei auf der Regulation von Genexpressionen, so dass das Genereg-Corpus speziell solche Relationen beinhaltet. Aus den daraus resultierenden 32.155 Dokumenten wurden zufällig 314 für eine manuelle Annotation durch Experten ausgewählt. Die annotierten Relationen zwischen Genen und Proteinen sind asymmetrisch. Das bedeutet, dass die Richtung der Interaktion im Genereg-Corpus, genauer der Regulation, ermittelt werden muss. Ähnlich wie für das LLL05-Corpus, das unten beschrieben wird, sind also Akteur und Ziel der PPI zu bestimmen. Das Corpus verfügt über 1225 positive sowie 10289 negative PPI-Beispiele. Das Verhältnis von positiven zu negativen Beispielen liegt damit im Genereg-Corpus beinah doppelt so hoch, wie es für das AIMed-Corpus der Fall ist.

Das **LLL05-Challenge-Corpus** wurde im Rahmen des Learning Language in Logic 2005 (LLL05) Workshops erstellt. Die Dokumente des Corpus sind der MEDLINE-Datenbank entnommen. Sie wurden durch die Suchanfrage „Bacillus subtilis and transcription“ gewonnen und beziehen sich im Schwerpunkt folglich auf entsprechende Phänomene. Es besteht eine Unterteilung in zwei Corpora, die für das Training gedacht sind, sowie ein Testcorpus[4]. Das erste Trainingscorpus enthält einfache PPI, während das zweite durch Koreferenzen einen höheren Schwierigkeitsgrad bieten soll. Teil des LLL05-Challenge ist es, Akteur und Ziel einer PPI zu identifizieren. Damit sind die zu extrahierenden Relationen nicht symmetrisch, sondern verfügen über eine Richtung. Es sind neben Satz- und Tokenannotationen auch die Lemmata der Tokens verfügbar sowie ein Link-Dependenzparse (Sleator und Temperley 1993). Ersteres Trainingscorpus kann im einheitlichen XML-Format von der gleichen Internetseite heruntergeladen werden, von der auch das AIMed-Corpus bezogen werden kann. Da es sich aber lediglich um einen Teil des Corpus' handelt und außerdem festgestellt wurde, dass ein großer Teil der Entitätenannotationen fehlt, wurde dieses Material nicht verwendet. Stattdessen wurde das Konvertierungsskript auf der Internetseite von Pyysalo u. a. (2008) modifiziert, das die LLL05-Daten in das einheitliche XML-Format überführt. Die beiden Trainingscorpora wurden dazu zusammengeführt und gemeinsam in das XML-Format konvertiert. Für die Erkennung der Entitäten im LLL05-Material dient ein Wör-

[3] `http://www.julielab.de/`

[4] Die Daten können von `http://genome.jouy.inra.fr/texte/LLLchallenge/` heruntergeladen werden.

terbuch, das alle in Frage kommenden Entitätennamen aufführt. Dieses Wörterbuch wird im modifizierten Konvertierungsprogramm eingelesen und zur Annotation der NEs verwendet. Das so konvertierte Trainingsmaterial enthält 163 positive und 300 negative Beispiele von PPI. Das herunterladbare LLL05-Testmaterial enthält keine PPI-Annotationen und kann somit nicht lokal ausgewertet werden. Für die Evaluation dient ein Webprogram, auf das von der gleichen Webseite aus zugegriffen werden kann, die das Corpus zum Herunterladen anbietet. Dieses Programm erwartet das verarbeitete Testcorpus im LLL05-Format und zeigt die Ergebnisse in Recall, Precision und F-Maß an.

KAPITEL 7

Umsetzung und Implementationsdetails

Die Kernoperatoren aus Kapitel 4 wurden für diese Arbeit reimplementiert. Auf diese Weise kann zentral durch eine einzelne Applikation auf sie zugegriffen werden. In Abschnitt 7.1 wird die Applikation beschrieben, die diese und weitere Methoden zusammenführt. Es waren außerdem einige Hilfswerkzeuge nötig, um alle geforderten Aufgaben zu erfüllen, von denen die wichtigsten in Abschnitt 7.2 kurz vorgestellt werden. In Abschnitt 7.3 werden Anmerkungen zur Implementierung der Methoden aus Kapitel 4 aufgeführt.

7.1 Das JREX-Projekt

Das JREX-Projekt (JULIE *Relation Extraction*) ist eine Java-Applikation für die Erkennung und Klassifikation von binären Relationen. Da das Projekt vor allem auf Methoden des maschinellen Lernens zurückgreift, gibt es keine Beschränkung auf PPI. Das entsprechende Datenmaterial vorausgesetzt, ist JREX in der Lage, ebenfalls andere Arten von Relationen zu verarbeiten. Das Programm wird von Ekaterina Buyko am JULIE LAB Jena im Rahmen ihrer Dissertation entwickelt.

Ursprünglich wurde in JREX vor allem ein Maximum Entropy (ME)-Ansatz für die Klassifikation von Relationen verwendet. Im Rahmen dieser Arbeit wurde die Möglichkeit hinzugefügt, ebenfalls SVMs in Verbindung mit den unterschiedlichen Kernoperatoren zu nutzen, die in Kapitel 4 vorgestellt wurden. Als Realisierung der Supportvektormaschine an sich wurde dabei das LIBSVM-Paket in der Version 2.86 verwendet (Chang und Lin 2001). Es implementiert Algorithmen zur SV-Klassifikation sowohl in C++ als auch in Java. Für diese Arbeit fand LIBSVM als Java-Bibliothek Verwendung. Die Bibliothek wurde dahingehend geändert, dass für eine trennende Hy-

perebene $\mathcal{H}$ die Länge des Normalenvektors $\boldsymbol{w}$ abgefragt werden kann. Diese Information ist für das Schwellwertverfahren von Bedeutung, das in Abschnitt 5.3 beschrieben wurde.

Es ist in JREX unter Zuhilfenahme dieser Methoden möglich, folgende Aufgaben durchzuführen:

- *Training* eines Modells auf gegebenem Datenmaterial und Sicherung des Modells für spätere Verwendung,
- *Vorhersage* (engl. *prediction*) auf gegebenem Datenmaterial auf Grund eines zuvor trainierten bzw. gelernten Modells,
- *Kreuzvalidierung* auf gegebenem Datenmaterial,
- *Optimierung* bezüglich der Parameter C (SVM-Strafparameter) und $\mathcal{T}$ (Schwellwert) sowie
- verschiedene Operationen bezüglich Gram- bzw. Kern-Matrizen.

Das von JREX verwendete Datenformat, um Texte zu repräsentieren, ist das von *Apache UIMA*[1] gegebene XMI- bzw. Common Analysis System (CAS)-Format. Die *Unstructured Information Management Architecture (UIMA)* ist ursprünglich ein IBM-Projekt, das unterdessen zu Apache-Incubator[2] gehört. UIMA stellt ein Framework für die *Strukturierung* unstrukturierter Informationen, wie etwa natürlichsprachliche Texte ohne zusätzliche Bearbeitung, zur Verfügung. Einzelne Aufgaben wie Satzerkennung, Tokenisierung oder die NER werden in Module, so genannte *Analysis Engines (AE)*, eingeteilt. Durch eine einheitliche interne Datenrepräsentation mittels so genannter *CAS-Objekte* können leicht Prozessierungs-Pipelines für die Lösung konkreter Problemstellungen konstruiert werden. Die in Abschnitt 3.1 genannten Schritte zur Textstrukturierung können etwa direkt in UIMA-Komponenten umgesetzt werden, wie es in Abbildung 7.1 veranschaulicht ist. Eine weitere Besonderheit des JREX-Projekts ist die flexible und umfassende externe Konfigurierbarkeit vieler Komponenten. Es können Einstellungen für die Behandlung von Daten vorgenommen werden, der konkrete Klassifikator- und Kernoperatortyp kann ausgewählt werden, die Parameter für Klassifikator und Kerne können eingestellt werden und weiteres. Die Einstellungen werden durch eine so genannte *Feature Konfiguration* vorgenommen. Eine Feature Konfiguration ist durch eine Datei gegeben, die pro Zeile ein Schlüssel-Werte-Paar enthält. Soll für einen konkreten Klassifikationslauf etwa der SVM-Strafparameter 0,5 verwendet werden, wird

[1] `http://incubator.apache.org/uima/`
[2] `http://incubator.apache.org/`

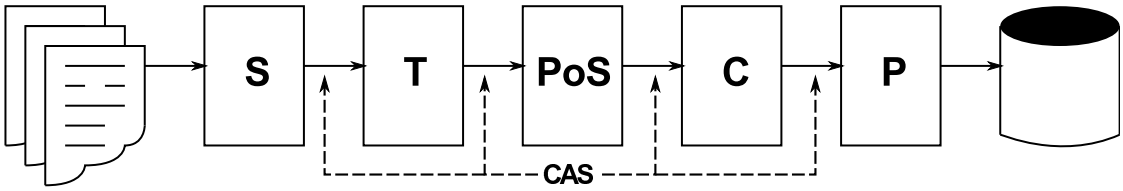

Abbildung 7.1: Schematische Abbildung einer UIMA Pipeline. Unstrukturierter Text (links) wird in strukturierte Form gebracht, etwa in Form einer Datenbank (rechts). Diese Abbildung zeigt je eine Analysis Engine zur Ausführung von **S**atzerkennung, **T**okenisierung, Wortartenerkennung (**PoS**), **C**hunking und **P**arsing. Zum einheitlichen Datenaustausch zwischen den AEs dienen so genannte CAS-Objekte.

in der Konfigurationsdatei die Zeile *'C = 0.5'* eingefügt. Die Datei wird JREX beim Programmstart übergeben und intern ausgelesen. Das ermöglicht neben einer flexiblen, persistent speicherbaren Konfiguration auch die automatische Exploration unterschiedlicher Einstellungen. Von dieser Möglichkeit wird etwa für die SVM-Parameteroptimierung Gebrauch gemacht, wie in Abschnitt 7.2 beschrieben wird.

Für den Parameter C der SVM und den Schwellwert $\mathcal{T}$ (vgl. Abschnitt 5.3) wird die Gittersuche verwendet, die in Abschnitt 5.2 beschrieben wurde. Die Gittersuche für diese beiden Parameter wurde direkt in das JREX-Projekt integriert, da die Parameter von der aufwändigen Berechnung der Kernwerte unabhängig sind. Durch die Integration ist es möglich, das Projekt zu starten, Daten einzulesen, Kernwerte bzw. Merkmale zu berechnen und anschließend mit diesen Informationen Kreuzvalidierungen mit unterschiedlichen Werten von C oder $\mathcal{T}$ durchzuführen. In Abschnitt 7.2 wird ein Programm beschrieben, das speziell für die Parameteroptimierung entwickelt wurde und auch für die Selektion von Parametern der Kernoperatoren eingesetzt werden kann. Da es sich dabei aber um ein externes Programm handelt, muss JREX für jeden Parametersatz neu aufgerufen werden, was zusätzliche Zeit kostet.

Bei der Arbeit mit Kern-SVMs gehört die Berechnung der Gram-Matrix zu den aufwändigsten Teilen der Klassifikation bezüglich der benötigten Rechenzeit. Um den Aufwand möglichst gering zu halten, nutzt JREX die Symmetrie der Gram-Matrizen aus und berechnet nur die obere Dreiecksform. Das heißt, eine Gram-Matrix G ist direkt nach der Werteberechnung von der Gestalt

$$G = \begin{pmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,m} \\ 0 & g_{2,2} & \cdots & g_{2,m} \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & 0 & g_{m,m} \end{pmatrix}.$$

Es ist möglich, die Gram-Matrix in dieser Form als Datei zu sichern und später

wieder zu laden und zu verwenden. Dadurch fällt für viele Experimente die Neuberechnung der Kernwerte weg. Sind zwei oder mehr Gram-Matrizen gespeichert, so können diese zu einer neuen, gültigen Gram-Matrix gemäß Gleichung 3.3.14 kombiniert werden. Dazu wird JREX eine Liste der zu kombinierenden Matrizen übergeben, wobei die Matrizen abgespeichert vorliegen müssen. Um die Berechnung der Gram-Matrix innerhalb von JREX weiter zu beschleunigen, ist die Erstellung der Matrix außerdem parallelisiert.

Im praktischen Kontext wird häufig vorgeschlagen, Gram-Matrizen zu normalisieren. Für die Normalisierung werden einige leicht unterschiedliche Formeln vorgeschlagen, die aber sehr ähnliche oder sogar die gleichen Auswirkungen haben. Normalisierung kann je nach Fall Leistungsverbesserungen bringen. Das ist allerdings nicht notwendigerweise der Fall, es kommt ebenfalls sogar vor, dass die Ergebnisse unter der Normalisierung der Gram-Matrix leiden. Deshalb ist es in JREX eine zusätzliche Option, eine Gram-Matrix-Normalisierung vorzunehmen. Die Normalisierbarkeit eines Kernoperators wurde bereits in Unterabschnitt 3.3.3 erläutert. In JREX wird die dort vorgestellte Gleichung 3.3.15 der Gestalt

$$\tilde{K}(\boldsymbol{x}, \boldsymbol{y}) = \frac{K(\boldsymbol{x}, \boldsymbol{y})}{\sqrt{K(\boldsymbol{x}, \boldsymbol{x}) K(\boldsymbol{y}, \boldsymbol{y})}}$$

verwendet, falls eine Normalisierung des verwendeten Kernoperators gewünscht ist.

7.2 Hilfswerkzeuge

In diesem Abschnitt werden die Hilfsprogramme vorgestellt, die für die Bearbeitung der Aufgaben nötig waren, die mit dieser Arbeit verbunden sind. Es handelt sich dabei um eine Anwendung für die Parametersuche, Konvertierungsskripte, um unterschiedliche Formate von Corpora ineinander zu überführen und Programme für die Schätzung von Wahrscheinlichkeitsverteilungen für die Kernoperatoren, die entsprechende Informationen verwenden. Alle in diesem Abschnitt beschriebenen Programme wurden mit Java in der Version 1.6 entwickelt und verwendet.

In Abschnitt 5.2 wurde die Gittersuche vorgestellt, mit deren Hilfe gute **Parameter für die Klassifikation** von PPI bestimmt werden sollen. Um diese Strategie umzusetzen, wurde das Werkzeug ParameterOptimizer entwickelt. Das Programm ist extern und vom JREX-Projekt zunächst unabhängig, auch wenn es bislang ausschließlich mit diesem verwendet wird. Der ParameterOptimizer ist in der Lage, eine *Featurekonfigurationsdatei*, wie sie in Abschnitt 7.1 beschrieben wurde, zu lesen und auch neu zu erstellen und wieder zu sichern. Dadurch kann eine vorhandene Konfiguration als Schablone

verwendet werden und es ist möglich, alle Einstellungen zu steuern, die sich über eine Konfigurationsdatei beeinflussen lassen. Damit sind Einstellungen über bloße SVM- oder Kernparameter hinaus möglich.

Es werden für numerische sowie boolesche Parameter untere und obere Schranken sowie Schrittweite und -art der Wertveränderung angegeben. Gibt man etwa für den Parameter C eine untere Schranke von 2^{-10} und eine obere Schranke von 2^8 sowie eine exponentielle Wertveränderung an, so werden die Werte $2^{-10}, 2^{-9}, \ldots, 2^8$ für das Feature C in der Konfigurationsdatei angegeben. Tatsächlich wird für jede *Wertekombination* von Parametern eine eigene Konfigurationsdatei erstellt, die sukzessive mit JREX verwendet werden, um die entsprechenden Parameter zu untersuchen. Wertekombinationen werden dann erstellt, wenn für mehrere Parameter die obigen Informationen, obere sowie untere Schranke, Schrittweite und -art, angegeben werden. Standardmäßig werden alle möglichen Kombinationen - also das gesamte durch die Werteangaben induzierte Gitter - erstellt und schrittweise auf ihre Leistungsauswirkung untersucht. Es ist allerdings möglich anzugeben, dass bestimmte Parameter stets den gleichen Wert erhalten sollen, um den Suchraum so gut wie möglich anpassen und gegebenenfalls verkleinern zu können. Das ist bei einer großen Anzahl von Parametern schnell nötig, da die benötigte Zeit für die Parameterexploration mit der Anzahl der Parameter kombinatorisch wächst. Deshalb ist der ParameterOptimizer parallel formuliert und in der Lage, beliebig viele Threads einzusetzen. Jeder Thread entspricht dabei einer Parameterkombination, mit der er einen eigenen JREX-Prozess startet. Das JREX-Projekt gibt dann die erzielten Resultate durch eine *Socket-Verbindung* an den ParameterOptimizer zurück. Dieser startet daraufhin die Untersuchung der nächsten ausstehenden Parameterkombination.

Durch die Möglichkeiten, die das Programm bietet, kann die Konfiguration schnell komplexe Ausmaße annehmen. Deshalb wurde für das Werkzeug eine grafische Oberfläche entwickelt, über die Konfiguration und Parametersuche möglich sind. In dieser Oberfläche werden alle Einstellungen vorgenommen und anschließend gespeichert, so dass sie später wieder geladen und angepasst werden können. Da die Suche mitunter allerdings sehr lange dauert, ist es auch möglich, eine Einstellungsdatei ohne Verwendung der grafischen Oberfläche direkt zur Parametersuche zu verwenden. Das gestattet etwa die Verwendung zusammen mit dem Programm 'screen', durch das ein Prozess aufrecht erhalten werden kann, ohne dass der Benutzer im Betriebssystem angemeldet bleiben muss.

Für die Arbeit mit verschiedenen Datenformaten wurden einige Format-**Konvertierungswerkzeuge** geschrieben. Pyysalo u. a. (2008) haben einige PPI-Corpora in ein einheitliches XML-Format gebracht, so dass nur eine einzelne Anwendung zur

Konversion dieses Formates in das von JREXverwendete Format nötig ist. Ein entsprechendes Programm, der PPI-Corpora-Reader von Oleg Lichtenwald am JULIE LAB Jena, war mit Grundfunktionalitäten vorhanden, musste aber bezüglich eines neuen XML-Formats der Corpora und der Extraktion von Corpuselementen wie Parse- und Part-of-Speech (PoS)-Informationen angepasst werden. Das LLL05-Corpus wurde ebenfalls von Pyysalo u. a. (2008) im XML-Format bereitgestellt. Allerdings wurde bei der Konversion offenbar übersehen, dass im Originalcorpus nicht alle Entitäten explizit gekennzeichnet sind. Zum Zweck der NER liegt dem LLL05-Corpus ein Wörterbuch bei, das alle Entitätennamen enthält, die im Corpus Erwähnung finden. Das Konvertierungsskript, das von Pyysalo u. a. (2008) verwendet wurde, um den LLL-Corpus in das von ihnen vorgestellte, einheitliche XML-Format zu überführen, lässt sich an gleicher Stelle wie die konvertierten Corpora öffentlich herunterladen[3]. Es handelt sich dabei um eine Python-Applikation, die in ihrer Originalversion keinen Gebrauch von dem erwähnten Wörterbuch macht. Das Script identifiziert Entitäten anhand ihrer Beteiligung an einer Interaktion. Entsprechend fehlen viele Negativbeispiele, also Entitätenpaare zwischen denen keine gesuchte Relation besteht. Das Konvertierungsskript wurde angepasst, so dass auch die Negativbeispiele generiert werden. Zudem wurde es so erweitert, dass es auch in der Lage ist, das LLL05-Testmaterial in das XML-Format zu konvertieren, so dass darauf gearbeitet werden kann.

Da das LLL05-Testmaterial keine PPI-Annotationen enthält, kann nicht direkt darauf evaluiert werden. Für das in Kapitel 6 erwähnte Web-Evaluierungsprogramm müssen die Klassifikationsergebnisse in das LLL05-Format überführt werden. Für diesen Zweck wurde ein Programm geschrieben, dass die Vorhersagen auf dem Testmaterial in das gewünschte Format konvertiert.

Für die Arbeit mit dem Verteilungskern und dem Local-Alignment-Kern werden **Schätzungen von Kookkurrenzwahrscheinlichkeiten** benötigt. Dafür wurde die TREC06 Genomics Textsammlung (Hersh u. a. 2006) heruntergeladen. Diese Sammlung besteht aus 162.259 Texten im HTML-Format aus dem Feld der Biomedizin. Um die HTML-Formate für die Wahrscheinlichkeitsschätzung zu verwenden, wurde ein Unstructured Information Management Architecture (UIMA)-Reader geschrieben, der die TREC06-Dokumente einlesen und in CAS-Objekte konvertieren kann. Dabei werden HTML-Tags sowie der Referenzteil der Dokumente entfernt. Zwei UIMA-Analysis Engines (AE) arbeiten jeweils auf den so bearbeiteten TREC06-Texten. Die erste AE dient zur Zählung der häufigsten Wörter (gegeben eine eine Liste von Stoppwörtern, die nicht berücksichtigt werden sollen), um ein Kookkurrenzvokabular aufzubauen. Dieses Vorgehen wurde von Séaghdha und Copestake (2008) vorgeschlagen. Die zweite

[3] http://mars.cs.utu.fi/PPICorpora/

AE berechnet auf Grund eines so gewonnenen Kookkurrenzvokabulars die Verteilungen für diese Worttypen bezüglich einer gegebenen Relation. Als Relationen kommen dabei Nachbarschaftsrelationen einer festen Nachbarschaftsgröße in Frage als auch Relationen, die durch einen Dependenzparse bestimmt werden. Es wird folglich gezählt, wie oft ein Worttyp mit einem anderen Typ durch eine Dependenzkante mit einer bestimmten Bezeichnung verbunden wird.

7.3 Kernoperatoren und Merkmalextraktion

Bei der Realisierung mancher Konzepte aus Kapitel 4 stellten sich praktische Schwierigkeiten oder es kamen Fragen auf, die durch die Originalbeschreibung des Verfahrens nicht hinreichend geklärt wurden. Dieser Abschnitt kommentiert für entsprechende Methoden die Änderungen oder Erweitungen, die vorgenommen wurden.

7.3.1 Local-Alignment-Kern

Die Definition des Local-Alignment-Kerns in Abschnitt 4.8 beinhaltete die Berechnung der Smith-Waterman-Ähnlichkeit zwischen zwei Dependenzpfaden x und y und die anschließende Aufsummierung aller möglichen lokalen Ausrichtungen. Ursprünglich wurde der Kernoperator von Saigo u. a. (2004) formuliert und anschließend von Katrenko und Adriaans (2008) für die Verwendung in Zusammenhang mit der PPI-Extraktion angepasst. Saigo u. a. (2004) geben einen Algorithmus für die Implementierung des Kerns an, der die Schritte der Maß-Berechnung und anschließende Aufsummierung zusammenlegt. Er ist in Abbildung 7.2 gegeben. Es existiert eine öffentliche Implementierung dieses Algorithmus' in der Programmiersprache C[4]. Er wurde nach Java 1.6 portiert und so angepasst, dass er mit einer Parametermatrix von Kookkurrenzwahrscheinlichkeiten verwendet werden kann.

Diese Parametermatrix wird in der Originalimplementierung von Katrenko und Adriaans (2008) dynamisch für jeden Klassifikatorlauf erstellt. Es wird die TREC06 Sammlung zusammen mit einem System verwendet, das auf Grundlage eines gegebenen Wortes automatisch relevante Passagen bezüglich dieses Wortes aus der Sammlung extrahiert. Auf den so erhaltenen Passagen wird die Kookkurrenzverteilung für das betrachtete Wort geschätzt. Dieses Verfahren wird für alle Wörter in den kürzesten Dependenz-Argumentpfaden, die im aktuell betrachteten Datenmaterial vorkommen, durchgeführt. Auf diese Weise liegen für alle in Frage kommenden Wörter Wahrschein-

[4] Die Software ist von http://sunflower.kuicr.kyoto-u.ac.jp/ hiroto/project/homology.html herunterladbar.

1. Initialisierung: Für $i = 0, \ldots, |\boldsymbol{x}|$ und $i = 0, \ldots, |\boldsymbol{y}|$:

$$\begin{aligned}
\mathcal{M}(i,0) &= \mathcal{M}(0,j) = 0, \\
\mathcal{X}(i,0) &= \mathcal{X}(0,j) = 0, \\
\mathcal{Y}(i,0) &= \mathcal{Y}(0,j) = 0, \\
\mathcal{X}_2(i,0) &= \mathcal{X}_2(0,j) = 0, \\
\mathcal{Y}_2(i,0) &= \mathcal{Y}_2(0,j) = 0,
\end{aligned}$$

   wobei $|\boldsymbol{x}|$ die Länge der Sequenz $\boldsymbol{x}$ bezeichne.

2. Die Summierung über alle möglichen lokalen Ausrichtungen wird in das dynamische Programm für die Berechnung des Smith-Waterman-Maßes integriert. Für $i = 1, \ldots, |\boldsymbol{x}|$ und $i = 1, \ldots, |\boldsymbol{y}|$:

$$\begin{aligned}
\mathcal{M}(i,j) &= e^{\beta \cdot d(x_i, y_i)} \left(1 + \mathcal{X}(i-1,j-1) + \mathcal{Y}(i-1,j-1) + \mathcal{M}(i-1,j-1)\right), \\
\mathcal{X}(i,j) &= e^{\beta \cdot l} \mathcal{M}(i-1,j) + e^{\beta \cdot e} \mathcal{X}(i-1,j), \\
\mathcal{Y}(i,j) &= e^{\beta \cdot l} \left(\mathcal{M}(i,j-1) + \mathcal{X}(i,j-1)\right) + e^{\beta \cdot e} \mathcal{Y}(i,j-1), \\
\mathcal{X}_2(i,j) &= \mathcal{M}(i-1,j) + \mathcal{X}_2(i-1,j) \\
\mathcal{Y}_2(i,j) &= \mathcal{M}(i,j-1) + \mathcal{X}_2(i,j-1) + \mathcal{Y}_2(i,j-1).
\end{aligned}$$

   Mit dem Parameter $\beta > 0$, den Strafparametern für Löschung bzw. Einfügung l und e sowie der Austauschparametermatrix $d(\cdot,\cdot)$.

3. Terminierung:

$$K_{\mathrm{LA}}^{(\beta)}(\boldsymbol{x}, \boldsymbol{y}) = 1 + \mathcal{X}_2(|\boldsymbol{x}|, |\boldsymbol{y}|) + \mathcal{Y}_2(|\boldsymbol{x}|, |\boldsymbol{y}|) + \mathcal{M}(|\boldsymbol{x}|, |\boldsymbol{y}|)$$

Abbildung 7.2: Ein Algorithmus, der direkt den Wert des Local-Alignment-Kerns zweier Sequenzen $\boldsymbol{x}$ und $\boldsymbol{y}$ berechnet.

lichkeitsschätzungen vor. Im Rahmen dieser Arbeit musste aus Gründen des zeitlichen Aufwandes auf ein solches so genanntes *Passage Retrieval System* verzichtet werden. Séaghdha und Copestake (2008) schlagen für die Bildung eines Kookkurrenzvokabulars vor, in der Textsammlung, die zur Verteilungsschätzung dienen soll, die n häufigsten Wörter zu suchen und diese als Vokabular zu verwenden. Im Bereich der Biomedizin stellt sich bei diesem Ansatz wieder das Problem, dass Gen- und Proteinnamen in ihrer Gestalt äußerst divers auftreten und ihre Zahl sehr hoch ist. Ein solches Vorgehen würde deshalb vermutlich viele Wörter von Interesse nicht berücksichtigen. Ein zweiter Ansatz besteht darin, alle Wörter auf kürzesten Entität-Entität-Pfaden im zu bearbeitenden Material als Kookkurrenzvokabular zu verwenden. Dies kommt dem Originalansatz näher, da auf die Wörter im Material zurückgegriffen wird. Allerdings ist die Schätzung nicht so differenziert, wie sie sich unter Verwendung eines Passage Retrieval Systems gestaltet, da dort Angaben etwa über die Zahl der extrahierten Passagen gemacht werden können. Zudem ist bei diesem Ansatz vor jedem Klassifikationslauf auf unbekanntem Datenmaterial eine vorhergehende Schätzung von Wahrscheinlichkeiten nötig.

Die letztlich verwendete Methode für die Schätzung der Kookkurrenzwahrscheinlichkeiten gestaltet sich folgendermaßen: Es wird zunächst ein Grundvokabular aus den Tokens der kürzesten Pfade des Datenmaterials gebildet. Die Wahrscheinlichkeitsschätzung findet auf der TREC06-Sammlung statt. Für alle Tokens t_g im Grundvokabular, die in der Sammlung gefunden werden, wird eine Wortnachbarschaft von 2 Tokens vor sowie nach t_g betrachtet. Diese Wortnachbarschaft wird durch eine Stoppwortliste gefiltert, die Wörter enthält, die sehr oft vorkommen, aber mit sehr hoher Wahrscheinlichkeit nicht von Interesse sind. Alle Nachbarschaftswörter von t_g, die nicht in der Stoppwortliste enthalten sind, werden in das Kookkurrenzvokabular eingetragen. Es werden für diese Wörter aber keine Wahrscheinlichkeitsverteilungen berechnet - sie dienen lediglich dazu, die Wörter bzw. Tokens t_g zu charakterisieren. Es wird eine obere Grenze angegeben, die das Kookkurrenzvokabular annehmen darf. Neue Wörter, die nach Erreichen dieser Grenze beobachtet werden, werden nicht berücksichtigt. Dieses Vorgehen stellt einen Kompromiss aus oben skizzierten möglichen Verfahren dar und konzentriert sich dabei darauf, die Wörter von Interesse zu beschreiben. Würde man einfach nur die n häufigsten Wörter aus der TREC06-Sammlung verwenden, würde man zum Einen unbenötigte Verteilungen schätzen und andererseits Verteilungen für interessante Wörter verlieren. Die Verteilungen werden nach dem Maximum-Likelihood-Prinzip geschätzt, stellen also ungeglättete, relative Häufigkeiten dar. Um die Datendichte hoch zu halten, werden alle Wörter des Kookkurrenzvokabulars zudem auf ihren Wortstamm reduziert.

Die Betrachtungen bezüglich der Art und Weise der Schätzung von Kookkurrenzwahrscheinlichkeiten treffen ebenfalls auf den Verteilungskern aus Abschnitt 4.7 zu.

7.3.2 Graph-Kern

Die Originalsoftware, die den Graph-Kern aus Airola u. a. (2008) implementiert, ist frei verfügbar[5]. Diese Software wurde näher studiert, um Fragen bezüglich Implementationsdetails zu klären. Dabei fielen zwei Besonderheiten auf, die sich auf die Implementierung für diese Arbeit ausgewirkt haben.

In Abschnitt 4.9 wurde ausgeführt, wie aus den Adjazenzmatrizen zweier Graphen durch mathematische Operationen ein Kernwert gewonnen werden kann. Im Zuge dessen wurde die Neumann-Reihe W einer ursprünglichen Adjazenzmatrix A gebildet. Diese wurde schließlich durch Links- sowie Rechtsmultiplikation der Labelallokationsmatrix L auf eine einheitliche Dimension gebracht. Der letzte Schritt ist in Gleichung 4.9.1 gezeigt und hat die Gestalt

$$G = LWL^\top.$$

Die Matrix L besitzt die Dimension $|\mathcal{L}| \times |\mathcal{V}|$, wobei $\mathcal{L}$ alle möglichen Labels bezeichnet und damit potentiell sehr groß ist. Daher wird L für obige Berechnung weder explizit dargestellt noch werden die beiden damit verbundenen Multiplikationen nach den herkömmlichen Methoden durchgeführt. Da L eine Binärmatrix ist - und demnach nur Nullen und Einsen enthält - stellt sie einen Spezialfall dar, der für eine effiziente Implementierung genutzt werden kann.

Wird die Binärmatrix L von links an die Matrix W heranmultipliziert, so entspricht das der Auswahl von Zeilen aus W: Enthält L an der Stelle $[L]_{ij}$ eine 1, so wird für die i-te Zeile der Resultatmatrix R die j-te Zeile von W ausgewählt. Enthält L in einer Zeile mehrere Einträge des Wertes Eins, so werden die entsprechenden Zeilen aus W Elementweise aufsummiert. Man stelle sich die Zeilen aus W dazu als transponierte Vektoren $v_1^\top, v_2^\top, \ldots, v_n^\top$ vor. Die Zeile i der Resultatmatrix $R = LW$ ergibt sich dann aus

$$\sum_{\{\, j | [L]_{ij} = 1 \,\}} v_j.$$

Der Fall, dass L von rechts an W heranmultipliziert wird, gestaltet sich analog. Es werden allerdings Spalten aus W an Stelle von Zeilen ausgewählt.

[5] Herunterladbar von http://mars.cs.utu.fi/PPICorpora/GraphKernel.html.

Durch die beidseitige Multiplikation der gleichen Binärmatrix ergibt sich eine Kombination aus Zeilen- und Spaltenauswahl. Genauer gilt für die Resultatsmatrix

$$[G]_{ij} = \sum_{\{\,(l,k):[L]_{ik}=1 \wedge [L^\top]_{kj}=1\,\}} [W]_{lk}. \tag{7.3.1}$$

Durch geschickte Indizierung der Labels lässt sich diese Formel mit einem Aufwand von $O(n^2)$ implementieren, wenn die Neumann Reihe W von der Dimension $n \times n$ ist.

Bei der näheren Untersuchung der Software von Airola u. a. (2008) fiel auf, dass an dieser Stelle keine Summe gebildet wurde. Stattdessen wurde der jeweils zuletzt auftretende Summand $[W]_{lk}$ der Stelle $[G]_{ij}$ zugewiesen und ein eventuell vorher bereits an diese Stelle geschriebener Wert wurde überschrieben. Durch E-Mail-Korrespondenz mit dem Hauptautor des Aufsatzes wurde bestätigt, dass ein Fehler im Originalprogramm vorliegt. Darauf folgende Experimente des verantwortlichen Autors ergaben, dass die Korrektur des Fehlers die Ergebnisse des Systems verschlechterte. Deshalb wurde dem Autor der vorliegenden Arbeit eine überarbeitete Version der oben beschriebenen Formel zugesandt. Von allen Summanden, die in Gleichung 7.3.1 vorkommen, wird in der neuen Version lediglich das Maximum verwendet, d.h.

$$[G]_{ij} = \max_{i \leq k,l \leq n} \{\, [L]_{ik}[W]_{kl}[L^\top]_{lj} \,\}. \tag{7.3.2}$$

Welche Version der Formel im Rahmen dieser Arbeit die besten Leistungen bringt, soll den Untersuchungen in Kapitel 8 überlassen werden.

Der zweite Punkt, der bei der Sicht der Originalsoftware auffiel und bei der Implementierung des Graph-Kerns hilfreich war, betrifft die Art und Weise, wie der kürzeste Pfad zwischen zwei Entitäten extrahiert wird. Tatsächlich beschränken sich Airola u. a. (2008) nicht auf einen einzelnen kürzesten Pfad, sondern verwenden alle kürzesten Pfade zwischen allen Tokens, die in den beiden Entitäten vorhanden sind[6]. Bestehen beide potentiellen Argumente lediglich aus einem Token, so ist der kürzeste Pfad eindeutig bestimmt. Es kommt aber vor, wie in Kapitel 2 und Abschnitt 5.1 bereits ausgeführt wurde, dass eine einzelne Entität so annotiert ist, dass sie mehrere Tokens umfasst. Die ursprüngliche Implementierung im Rahmen dieser Arbeit extrahiert genau einen kürzesten Pfad zwischen den Entitäten. Zu diesem Zweck wurde der Kopf der Tokengruppe einer Entität identifiziert und anschließend ein kürzester Pfad zwischen den beiden Köpfen der Argumentkandidaten gesucht. Bei Entitäten mit mehre-

[6]Der Autor dieser Arbeit fand im Algorithmus für die Extraktion aller kürzesten Pfade in der Software von Airola u. a. (2008) ebenfalls einen Fehler. Dieser verhinderte, dass in einigen Fällen gänzlich alle kürzesten Pfade geliefert wurden. Eine Korrektur dieses Problems hatte keine größeren Auswirkungen, da es sich wohl eher um einen Randfall handelte. Der Hauptautor des Aufsatzes wurde über den Fehler informiert.

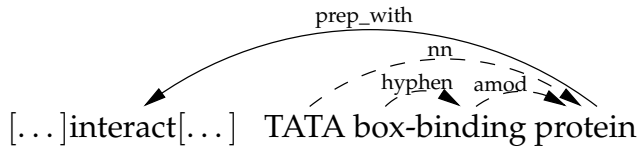

Abbildung 7.3: Die Entität *'TATA box-binding protein'* aus dem AIMed-Corpus. Um den Kopf der Entität zu finden, die aus mehreren Tokens besteht, wird das Token gesucht, dessen Dependenzkante aus der Entität herausführt.

ren Tokens ist es typischerweise so, dass fast alle Tokens der Entität auf andere Tokens innerhalb der Entität verweisen. Das Token, dessen Dependenzkante aus der Entität herausführt, wurde als Kopf der Entität definiert (vgl. Abbildung 7.3). Es ergab sich durch Experimente, dass die Verwendung aller Pfade zwischen den Tokens zweier Argumentkandidaten wesentlich für die Leistung des Graph-Kerns ist. Daher werden in der Implementierung für JREX ebenfalls alle kürzesten Pfade verwendet.

Zusätzlich wurde für alle Tokens eine Reduzierung auf den Wortstamm durchgeführt.

7.3.3 Walk-Merkmale

Kim u. a. (2008) beschreiben für die Erstellung von Walk-Merkmalen den Fall, dass ein Prädikat auf dem kürzesten Pfad zwischen zwei betrachteten Entitäten liegt. Zusätzlich wird davon ausgegangen, dass das Prädikat die Wurzel des durch den Pfad dargestellten Teilbaums bildet, was typischerweise auch der Fall ist. Es gibt allerdings weitere Erscheinungsbilder des kürzesten Pfades, die in Abbildung 7.4 illustriert sind. Da jedes Entitätenpaar als potentielles Argumentenpaar einer Relation angesehen wird, kommt es auch vor, dass kein Prädikat auf dem kürzesten Pfad existiert. Wie weiterhin in Abschnitt 3.1 erwähnt wurde, gibt es keine allgemeine Konvention dafür, ob in einer Dependenzrelation der Dependenzkopf auf den Dependenten zeigt oder anders herum. Die Kantenrichtung ist also nicht global festgelegt, sie ist nur für jedes konkrete Parsingschema in sich konsistent. Da Kim u. a. (2008) ausschließlich den Link-Parse berücksichtigen, der im LLL05-Corpus vorangelegt ist, muss dort auf diesen Punkt nicht weiter eingegangen werden. Da das Verfahren innerhalb der vorliegenden Arbeit allerdings allgemeiner eingesetzt werden soll, ist eine generellere Formulierung dafür nötig, wann eine Kante nach „oben“ und wann sie nach „unten“ zeigt. Für den Fall, dass kein Prädikat auf dem kürzesten Pfad enthalten ist, gibt es keinen Anhaltspunkt in Form einer Baumwurzel, an dem sich das Verfahren orientieren könnte. Bei verschachtelten Prädikat-Argument-Strukturen und anderen komplizierteren Abhängigkeitsstrukturen

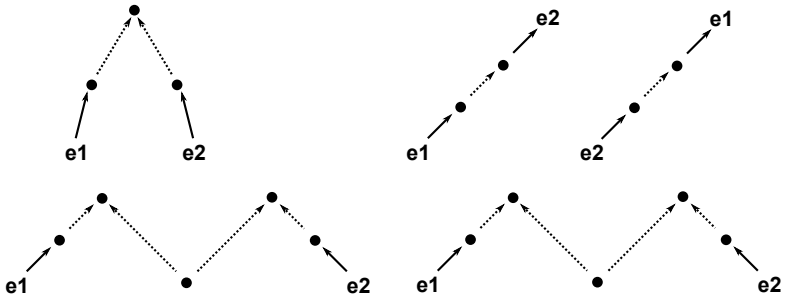

Abbildung 7.4: Unterschiedliche Erscheinungsbilder von kürzesten Pfaden zwischen zwei Entitäten. Im Fall, der oben links abgebildet ist, liegt ein Prädikat auf dem kürzesten Pfad. Dieser Fall wurde von Kim u. a. (2008) als Standardfall beschrieben. Rechts oben sind Fälle skizziert, bei denen kein Prädikat im Pfad enthalten ist. Es kommt außerdem vor, dass mehrere Richtungswechsel von Kanten in einem kürzesten Pfad enthalten sind, was in den unteren Skizzen gezeigt ist.

kommt es sogar vor, dass der Pfad mehrere Richtungswechsel vollführt. Für eine einheitliche Erfassung dieser unterschiedlichen Strukturen wurde vereinbart, dass eine Dependenzkante für die Walk-Merkmale den Zusatz *'UP'* erhält, falls die Kante gemäß ihrer Richtung traversiert wird. Wird entgegen der Kantenrichtung traversiert, erhalten die Merkmale die zusätzliche Zeichenkette *'DN'*. Zudem wird ein Richtungswechsel nicht nur in dem Fall angezeigt, der in Abbildung 7.4 oben links skizziert ist. Stattdessen wird jeder Richtungswechsel einheitlich als solcher markiert, das heißt es wird für alle Richtungswechsel die gleiche markierende Zeichenkette verwendet.

Motiviert durch die Strategie des Graph-Kerns, alle kürzesten Dependenzpfade zwischen den Tokens zweier Argumentkandidaten zu verwenden, wurde diese Strategie auch für die Walk-Merkmale getestet. Dieses Vorgehen verschlechterte die Ergebnisse in den meisten Experimenten allerdings signifikant. Daher wird ein einziger kürzester Pfad zwischen zwei Entiten extrahiert, der durch den kürzesten Pfad zwischen den Köpfen der Entitäten gegeben ist. Das Verfahren wurde in Unterabschnitt 7.3.2 im Kontext des Graph-Kerns bereits näher beschrieben. Auf gleiche Weise werden außerdem für den Verteilungskern die Köpfe der Entitäten ermittelt, auf denen daraufhin der zusammengesetzte Verteilungsvektor generiert wird.

Falls gar kein kürzester Pfad zwischen zwei Entitäten gefunden werden kann, so wird der entsprechende Relationskandidat ausgelassen. Genauer bedeutet das, dass der Kandidat im Falle des Trainings völlig ignoriert und bei der Vorhersage der Nicht-PPI-Klasse zugeordnet wird. Diese Vorgehensweise trifft analog auf den Local-Alignment-Kern und die Dependenzmerkmale zu, da diese Verfahren einen Dependenzpfad zwischen den Argumenten voraussetzen.

KAPITEL 8

Ergebnisse und Analyse

Für die Auswertung der in dieser Arbeit vorgestellten Methoden werden die Corpora heranzgezogen, die in Kapitel 6 vorgestellt wurden. Üblicherweise werden 10-fache Kreuzvalidierungen vorgenommen, die als Kompromiss zwischen zeitlichem Aufwand und Materialausnutzung dienen. Bevor konkrete Auswertungen durchgeführt werden, sollen zunächst einige Vorgehensweisen für die Evaluation festgelegt werden. Diese Richtlinien sorgen einerseits für einen klaren, geordneten Ablauf und reduzieren andererseits die Anzahl der Experimente auf ein übersichtliches Maß. Es gelten folgende Vereinbarungen:

- Alle Kernwerte werden gemäß Gleichung 3.3.15 normalisiert.
- Bei Parametersuchen werden zunächst die Parameter der Kernoperatoren zusammen mit dem SVM-Strafparameter C bestimmt. Der C-Parameter wird stets aus der Menge $\{ 2^{-5}, 2^{-4}, \ldots, 2^{10} \}$ ausgewählt. Erst im Anschluss wird mit den so ermittelten Parametern ein Schwellwert $\mathcal{T}$ für die Erhöhung des Recalls aus der Menge $\{ 2^{-10}, 2^{-9}, \ldots, 2^{-1} \}$ ermittelt.
- Merkmalansätze werden mit dem linearen Kernoperator $\langle \cdot, \cdot \rangle$ verwendet sowie mit dem RBF-Kern expandiert, der durch Gleichung 3.3.13 gegeben ist. Der Parameter γ des RBF-Kerns wird aus der Menge $\{ 2^{-10}, 2^{-9} \ldots, 2^{2} \}$ selektiert.
- Es wird stets - in Parametersuche und in der folgenen Methodenauswertung - ein bester Wert des F-Maßes gesucht.
- Für die Schätzung aller verwendeten Verteilungen für den Verteilungskern und den Local-Alignment-Kern wird eine Stoppwortliste verwendet; außerdem wird nur die Nachbarschaftsrelation mit einem Fenster von 2 Tokens links und rechts des Tokens von Interesse verwendet. Es werden Kookkurrenzvokabulare einer Größe von 10.000 Wörter in Betracht gezogen.

Formel	AIMed	Genereg
Original	$R: 48,20$, $P: 57,44$, $F: 52,41$	$R: 50,28$, $P: 67,54$, $F: 57,65$
Maximum	$R: 48,10$, $P: 60,96$, $\mathbf{F}: \mathbf{53,77}$	$R: 51,67$, $P: 72,67$, $\mathbf{F}: \mathbf{60,40}$

Tabelle 8.1: Ergebnisse des Experiments bezüglich der Formel zur Berechnung der Matrix G des Graph-Kerns.

Es muss entschieden werden, ob allgemein eine Kernwertnormalisierung durchgeführt werden soll oder nicht. Eine Festlegung soll der Vergleichbarkeit der Ergebnisse dienen, da die Normalisierung nicht als Optimierungsfaktor gesehen werden soll. Es handelt sich dabei um eine Transformation des Merkmalsraums und es ist nicht klar, inwiefern Ergebnisse normalisierter und nicht normalisierter Kernwerte vergleichbar sind. Es wurde entschieden, alle Werte zu normalisieren, da sich dieses Vorgehen potentiell positiv auf das Schwellwertverfahren sowie auf die Trainingszeiten der SVM auswirkt.

Man könnte den Schwellwert $\mathcal{T}$ auch in Kombination mit Kernparametern und SVM-C optimieren. Hierdurch würde sich der Parameterraum allerdings jeweils um eine Dimension erweitern, so dass die Suche nach einer guten Parameterkombination entsprechend länger dauern würde. Zudem handelt es sich bei dem Schwellwertverfahren um eine Nachverarbeitung. Entsprechend soll $\mathcal{T}$ in dieser Arbeit zuletzt und losgelöst von anderen Optimierungen festgelegt werden. Es werden außerdem oben genannte Mengen festgelegt, aus der bei einer Parametersuche die Parameterwerte selektiert werden. Die Parameterwerte werden in den folgenden Beschreibungen von Experimenten der Übersicht halber nicht explizit angegeben. Zudem orientieren sich die Experimente als Gütemaß am F-Maß. Es stellt das harmonische Mittel aus den Maßen Recall (Ausschöpfung) und Precision (Genauigkeit) dar, so dass keiner dieser Größen ein höheres Gewicht als der anderen zugesprochen wird.

Für die Schätzung von Verteilungen gibt es viele Detailfragen, die geklärt werden müssen. Da die Schätzung von Verteilungen auf Grund der Größe der TREC06-Sammlung zu den zeitintensivsten Aufgaben dieser Arbeit gehört, werden die oben genannten Einschränkungen getroffen.

Ein weiterer Punkt, der zu Beginn der Auswertung festgelegt werden soll, ist die Wahl der Formel für die Berechnung der Matrix G des Graph-Kerns. Wie in Unterabschnitt 7.3.2 ausgeführt wurde, liegen dem Autor dieser Arbeit zwei Formeln für die Berechnung der Matrix vor. Es wurden Experimente auf den beiden größeren Auswertungscorpora - also AIMed sowie Genereg - durchgeführt. Es wurde jeweils die Originalformel aus dem Aufsatz von Airola u. a. (2008) sowie die nachträglich entwi-

ckelte Formel aus Gleichung 7.3.2 verwendet. Die Ergebnisse dieses Vorversuchs sind in Tabelle 8.1 festgehalten. In diesen Experimenten erweist sich die neue Formel als performanter, so dass diese in allen folgenden Versuchen Anwendung findet, in denen der Graph-Kern einbezogen ist.

Zudem gestaltet sich die Parametersuche für den Local-Alignment-Kern als sehr aufwändig. Neben dem Strafparameter C müssen die freien Parameter für Löschung und Einfügung, sowie der β-Parameter belegt werden. Nach einigen kleineren Experimenten wurden diese Parameter auf je $0,2$ für Löschung und Einfügung und $\beta = 1,6$ festgelegt. Diese Werte sind an die Wahl von Katrenko und Adriaans (2008) angelehnt. Zudem wird das Cosinus-Maß eingesetzt, ebenso wie für den Verteilungskern[1]. Bei vielversprechenden Ergebnissen sollen auch andere Maße getestet werden.

Alle beschriebenen Verfahren wurden nach den eben festgelegten Grundsätzen auf allen Corpora ausgewertet. Dabei kamen stets 10-fache Kreuzvalidierungen zum Einsatz. Die Bag-of-Words-Merkmale sowie die zusätzlichen Vereinigungsmerkmale aus Abschnitt 4.6 werden jeweils in Verwendung mit dem linearen Kern betrachtet. Dies dient als Referenzpunkt für die Ergebnisse, die die Vereinigungen bzw. die Matrixkombinationen erzielen. Da diese Merkmale nicht als eigenständige Methoden zur PPI-Extraktion gedacht sind, werden sie aus detaillierten Betrachtungen und auch aus dem Schwellwertverfahren ausgenommen. Es werden zunächst alle Evaluationen vorgestellt, bevor die Methoden einer alle Corpora umfassenden Analyse unterzogen werden.

Die Ergebnisse für das **AIMed-Corpus** sind in Tabelle 8.2 aufgeführt, wobei noch kein Schwellwert $\mathcal{T}$ selektiert wurde. Das beste Ergebnis wurde durch Verwendung des Graph-Kerns erreicht und ist fett dargestellt. Das Mittelfeld bilden die Ergebnisse der Dependenzmerkmale zusammen mit denen der Walk-Merkmale. Während die zusätzlichen Merkmale, die für die Vereinigung gedacht sind, alleine ein eher schlechtes Resultat ergeben, hebt sich die Leistung der Vereinigung von Dependenz-, Walk- und den zusätzlichen Merkmalen deutlich von den einzelnen Merkmalansätzen ab. Auffällig ist, dass die Verwendung des RBF-Kerns die Ergebnisse konsequent verschlechtert. Die Resultate der Kerne, die Kookkurrenzverteilungen verwenden, stellen sich als am niedrigsten dar. Der Verteilungskern arbeitet wie in Unterabschnitt 7.3.1 vermutet offenbar nicht so, wie es gewünscht war. Aber auch das Ergebnis des Local-Alignment-Kerns bleibt weit hinter dem der Walk-Merkmale zurück, die ebenso auf dem kürzesten Dependenzpfad arbeiten. Auf Grund der schlechten Leistungen des Verteilungs- und Local-Alignment-Kerns wurden keine weiteren Untersuchungen bezüglich unter-

[1] In Abschnitt 4.7 wurde ausgeführt, dass das Cosinus-Maß dem L2-normierten, linearen L2-Kern entspricht.

Kern	Recall	Precision	*F*-Maß
Bag-of-Words-Merkmale (linear)	45,20	27,74	34,38
Zusätzliche Merkmale für die Vereinigung (linear)	29,10	35,74	32,08
Dependenzmerkmale (linear)	35,90	56,59	43,40
Dependenzmerkmale (RBF)	34,60	56,16	42,82
Walk-Merkmale (linear)	35,20	56,59	43,40
Walk-Merkmale (RBF)	34,60	56,16	42,82
Merkmalvereinigung (linear)	39,70	64,76	49,22
Merkmalvereinigung (RBF)	37,80	61,06	46,69
Verteilungskern	3,00	19,23	5,19
LA-Kern	24,20	22,76	23,46
Graph-Kern	**48,10**	**69,96**	**53,77**

Tabelle 8.2: Ergebnisse auf dem AIMed-Corpus ohne Schwellwertverfahren.

Kern	Recall	Precision	*F*-Maß
Dependenzmerkmale (linear)	47,70	33,63	39,45
Dependenzmerkmale (RBF)	37,60	36,64	37,11
Walk-Merkmale (linear)	47,40	41,87	44,46
Walk-Merkmale (RBF)	43,50	47,33	45,33
Merkmalvereinigung (linear)	**57,70**	46,38	51,42
Merkmalvereinigung (RBF)	48,60	50,67	49,61
Graph-Kern	55,20	**55,47**	**55,33**

Tabelle 8.3: Ergebnisse auf dem AIMed-Corpus nach durchgeführtem Schwellwertverfahren.

Kern	Recall	Precision	*F*-Maß
Bag-of-Words-Merkmale (linear)	43,01	13,12	20,11
Zusätzliche Merkmale für die Vereinigung (linear)	28,60	56,74	38,03
Dependenzmerkmale (linear)	24,13	31,80	27,44
Dependenzmerkmale (RBF)	21,84	39,31	28,08
Walk-Merkmale (linear)	30,87	51,08	38,48
Walk-Merkmale (RBF)	30,66	51,44	38,42
Merkmalvereinigung (linear)	34,42	60,31	43,83
Merkmalvereinigung (RBF)	29,87	65,82	41,10
Verteilungskern	0,24	8,33	0,47
LA-Kern	21,59	35,39	26,82
Graph-Kern	**51,67**	**72,67**	**60,40**

Tabelle 8.4: Ergebnisse auf dem Genereg-Corpus ohne Schwellwertverfahren.

schiedlicher Maße zum Vergleich von Wahrscheinlichkeitsverteilungen durchgeführt.

Wie zu Anfang des Kapitels beschrieben, wurde anschließend ein Schwellwertverfahren durchgeführt. Dabei wird der Schwellwert $\mathcal{T}$ sukzessive inkrementiert. Das hat zur Folge, dass zusätzliche Objekte als Interaktionen klassifiziert werden, die zuvor der Rückweisungsklasse zugewiesen wurden. Dieses Vorgehen erhöht die Ausschöpfung und senkt gleichzeitig die Genauigkeit. Tabelle 8.3 zeigt die Resultate des Verfahrens auf dem AIMed-Corpus. Der Verteilungs- und Local-Alignment-Kern wurden an dieser Stelle ausgenommen, da die Ergebnisse dieser Methoden für eine sinnvolle Nachverarbeitung nicht ausreichen. Das *F*-Maß des Graph-Kerns ist wieder das höchste, das erreicht wurde. Der Verfahren profitiert offenbar von der Erhöhung des Recalls. Insgesamt reagieren lediglich die Ergebnisse der Dependenzmerkmale negativ auf das Schwellwertverfahren. Diese Feststellung entspricht nicht den Erwartungen, da der niedrige Recall zuvor (siehe Tabelle 8.2) Verbesserungspotential erhoffen ließ.

Auf dem **Genereg-Corpus** wurde durch das gleiche Vorgehen evaluiert. Die Ergebnisse ohne Schwellwertverfahren sind in Tabelle 8.4 zusammengefasst. Die grobe Staffelung der Resultate gestaltet sich ähnlich zu denen des AIMed-Corpus'. Auffällig ist jedoch der geringe *F*-Maß-Wert der Dependenzmerkmale. Die Resultate der Verteilungs- und Local-Alignment-Kerne verbleiben in ähnlichen Größenordnungen wie bereits zuvor. Das signifikant beste Verfahren stellt der Graph-Kern dar, dessen Leistung auf dem Genereg-Corpus einen Abstand von 16,57% *F*-Maß zum Ergebnis der zweitbesten Methode erreicht. Weiterhin ist es wichtig zu bemerken, dass die meisten Ansätze auf dem Genereg-Corpus schlechtere Ergebnisse als auf dem AIMed-Corpus

Kern	Recall	Precision	*F*-Maß
Dependenzmerkmale (linear)	37,53	24,07	29,33
Dependenzmerkmale (RBF)	32,86	25,88	28,95
Walk-Merkmale (linear)	55,26	32,05	40,57
Walk-Merkmale (RBF)	42,95	41,19	42,05
Merkmalvereinigung (linear)	52,02	42,79	46,96
Merkmalvereinigung (RBF)	45,00	51,40	47,98
Graph-Kern	**70,04**	**57,23**	**62,99**

Tabelle 8.5: Ergebnisse auf dem Genereg-Corpus nach durchgeführtem Schwellwertverfahren.

hervorbringen. Der Wert des *F*-Maßes für den Graph-Kern liegt hingegen 6,63% vor dem Resultat, das auf AIMed erzielt wurde.

Es wurde ebenfalls ein Schwellwertverfahren durchgeführt, dessen Ergebnisse Tabelle 8.5 aufführt. Die beiden Kerne, die Verteilungen verwenden, wurden ausgelassen, da bereits zuvor nur sehr geringe Ergebnisse erzielt wurden. Hier reagieren die Resultate der Dependenzmerkmale durch einen leicht höheren Wert des *F*-Maßes, während alle anderen Methoden stärker profitieren. Insbesondere die Ergebnisse der Merkmale, die mit dem RBF-Kern eingesetzt wurden, werden besser. Dieses Verhalten ist ebenfalls auf dem AIMed-Corpus zu beobachten, dort allerdings nicht so ausgeprägt. Betrachtet man die vereinigte Merkmalmenge in Zusammenhang mit dem RBF-Kern, so gewinnt die Leistung des Verfahren 6,88% *F*-Maß hinzu. Es schließt damit ein wenig zum Resultat des Graph-Kerns auf, das allerdings durch das Schwellwertverfahren ebenfalls gewinnt. Der Ergebnisunterschied der beiden besten Verfahren beträgt 15,01% *F*-Maß.

Die Ergebnisse auf dem **LLL05-Corpus** sind in Tabelle 8.6 angegeben. Hier werden durch die Walk-Merkmale sehr gute Resultate erreicht. Ihre Leistung konnte durch die Vereinigung der Merkmalmengen aber offenbar leicht verbessert werden, wobei der Unterschied zu den Ergebnissen der Walk-Merkmalen mit linearem Kern sehr gering ist. Der *F*-Maß-Wert des Graph-Kerns liegt bei diesen Daten lediglich im oberen Mittelfeld. Sehr auffällig ist die Leistung des Local-Alignment-Kerns, das sogar über dem Ergebnis des Graph-Kerns liegt. Der Verteilungskern hingegen ergibt auch auf dem LLL05-Corpus kein nennenswertes Resultat.

Das Schwellwertverfahren führt ausschließlich zu Ergebnisverschlechterungen, wie Tabelle 8.7 zu entnehmen ist. Das ist bei den meisten Methoden nicht verwunderlich, da die Ausschöpfung auf dem LLL-Corpus allgemein hoch ist. Aber selbst das Resultat des Graph-Kerns, das zuvor über niedrigen Recall aber hohe Precision verfüg-

Kern	Recall	Precision	*F*-Maß
Bag-of-Words-Merkmale (linear)	61,22	60,00	60,60
Zusätzliche Merkmale für die Vereinigung (linear)	50,34	52,51	51,40
Dependenzmerkmale (linear)	57,14	64,23	60,48
Dependenzmerkmale (RBF)	50,67	55,97	53,19
Walk-Merkmale (linear)	**80,89**	76,50	78,63
Walk-Merkmale (RBF)	74,21	**81,37**	77,63
Merkmalvereinigung (linear)	78,98	77,98	**78,48**
Merkmalvereinigung (RBF)	72,95	78,37	75,57
Verteilungskern	0,61	25,00	1,19
LA-Kern	72,54	72,07	72,31
Graph-Kern	62,42	77,77	69,25

Tabelle 8.6: Ergebnisse auf dem LLL05-Corpus ohne Schwellwertverfahren.

Kern	Recall	Precision	*F*-Maß
Dependenzmerkmale (linear)	48,05	62,71	54,51
Dependenzmerkmale (RBF)	51,03	46,54	48,68
Walk-Merkmale (linear)	70,70	76,55	73,50
Walk-Merkmale (RBF)	**70,96**	**78,01**	**74,32**
Merkmalvereinigung (linear)	69,87	77,85	73,64
Merkmalvereinigung (RBF)	66,88	75,73	71,03
LA-Kern	64,08	49,45	55,82
Graph-Kern	65,10	66,43	65,76

Tabelle 8.7: Ergebnisse auf dem LLL05-Corpus nach durchgeführtem Schwellwertverfahren.

te, verschlechtert sich durch das Verfahren. Die Verwendung eines Schwellwertes ist an dieser Stelle offenbar nicht von Vorteil. Wie in Kapitel 6 erläutert wurde, besteht das LLL05-Corpus unter anderem aus einem Satz von Testmaterial, für den die richtigen Ergebnisse nicht bekannt sind. Die Methoden, die bei den 10-fachen Kreuzvalidierungen auf dem LLL05-Trainingsmaterial die besten Resultate ergaben, wurden für die Auswertung im Internet herangezogen. Die Resultate sind in Tabelle 8.8 zu sehen. Zunächst fallen die allgemein niedrigen Ergebnisse auf. Weiterhin liegt die Leistung der Merkmalvereinigung hier vor der der Walk-Merkmale, die zuvor scheinbar den Großteil der Ergebnisse auszumachen schien. Der *F*-Maß-Wert des Local-Alignment-Kerns liegt nur knapp hinter dem der Merkmalvereinigung. Zum Vergleich wurden die Ergebnisse der Originalimplementierungen von Walk-Merkmalen sowie des Local-

Kern	Recall	Precision	*F*-Maß
Walk-Merkmale (linear)	57,40	32,20	41,30
Merkmalvereinigung (linear)	59,20	**62,70**	**60,90**
LA-Kern	**61,10**	55,90	58,40
Orig. Walk-Merkmale (Kim u. a. 2008)	83,30	72,50	77,50
Orig. LA-Kern (Katrenko und Adriaans 2008)	50,00	71,00	58,60

Tabelle 8.8: Ergebnisse der Webevaluation des LLL05-Challenge. Es wurden die drei Verfahren ausgewählt, die bei der Kreuzvalidierung am besten abschnitten. Im unteren Teil der Tabelle sind die Ergebnisse aus den Originalaufsätzen über die Walk-Merkmale bzw. des Local-Alignment-Kerns angegeben.

Alignment-Kerns angegeben. Das Resultat der Walk-Merkmale im Original liegt weit vor dem der hier verwendeten Implementierung. Das ist einerseits durch die teilweise allgemeineren Formulierungen der Methodik zu erklären, die für den Gebrauch mit den anderen Corpora vorgenommen wurde (vgl. Unterabschnitt 7.3.3). Zudem wurde die Methode von Kim u. a. (2008) ausschließlich auf dem LLL05-Material ausgewertet, so dass von einer höheren Anpassung an das Material ausgegangen werden kann.

Das Resultat des Local-Alignment-Kerns ist dem Original insgesamt ähnlich. Dennoch ist die unterschiedliche Verteilung von Recall und Precision klar zu sehen. Dies ist vermutlich den sehr unterschiedlichen Arten und Weisen der Verteilungsschätzung geschuldet. Insgesamt scheint es, dass von den Leistungen der Kreuzvalidierungen auf den Trainingsdaten in diesem Fall nicht auf die Ergebnisse auf dem Testset geschlossen werden kann. Das könnte mit der geringen Größe des Gesamtmaterials zusammen hängen.

Zuletzt soll die Kombination von Kernwerten betrachtet werden. Dazu werden die beiden Methoden betrachtet, die die besten Resultate hervorgebracht haben. Bei dieser Bewertung wird das LLL05-Corpus außen vor gelassen, da es verglichen mit den anderen Corpora sehr klein ist. Durch die geringe Größe des Corpus ist wenig Material zum Lernen verfügbar, so dass die Ergebnisse weniger repräsentativ sind. Da das Schwellwertverfahren eine Nachprozessierung darstellt, werden die Werte vor der Verwendung eines Schwellwerts betrachtet. Ebenso werden für die Kombinationen keine Schwellwerte ermittelt, da es an dieser Stelle nur um die prinzipiellen Auswirkungen der Kombination gehen soll. Es wurden der Graph-Kern sowie die Merkmalvereinigung unter Verwendung des linearen Kerns ausgewählt. Es stellt sich bei dieser Wahl allerdings das Problem, dass die Vereinigung von Merkmalen nicht für alle Relationskandidaten eingesetzt werden kann. Ein solcher Fall ergibt sich bei potentiellen Rela-

Verfahren	AIMed	Genereg	LLL05
Graph-Kern mit Auslassung	51,82	60,48	70,30
Graph-Kern und Merkmalvereinigung	49,22	61,08	77,63
Graph-Kern und Bag-of-Words-Merkmale	32,08	59,20	70,70
Bestes Verfahren	53,77	60,40	78,37

Tabelle 8.9: Ergebnisse von Kernwert-Kombinationen. Alle Resultate sind als *F*-Maß angegeben. Der untere Teil der Tabelle zeigt eine Übersicht der Resultate des besten Verfahrens für jedes Corpus.

tionen, zwischen deren Argumenten kein Dependenzpfad gefunden werden kann. In diesem Fall können weder Walk- noch Dependenzmerkmale erstellt werden. Da entsprechende Kandidaten nicht in das Training eingehen, ist die Kernmatrix der Merkmalvereinigung von kleinerer Dimension als die Kernmatrix des Graph-Kerns. Der Graph-Kern kann immer eingesetzt werden, da der kürzeste Pfad für eine Wertberechnung nicht zwingend vorhanden sein muss. Um die Kombination dennoch durchführen zu können, werden auch für den Graph-Kern die Relationskandidaten ausgelassen, für die keine Merkmale generiert werden können. Da sich dieses Vorgehen auf das Ergebnis des Graph-Kerns ohne Kombination bereits auswirkt, sind die entsprechenden Ergebnisse zusammen mit den Resultaten der Kombination in Tabelle 8.9 angegeben. Aus dem Grund der potentiellen Unvereinbarkeit zweier Methoden wurden die Bag-of-Words-Merkmale eingeführt. Da sie auf keine speziellen Informationen zurückgreifen, können diese Merkmale stets angewendet werden.

Offenbar kann nicht allgemein davon ausgegangen werden, dass die vorgenommene Kombination der Methoden von Vorteil ist. Die Ansätze scheinen sich eher gegenseitig zu stören, so dass die Ergebnisse der Kombination beinah für alle Corpora hinter dem Resultat des einzelnen, besten Verfahren für jedes Corpus zurückbleiben. Lediglich das Auswertungsergebnis des Genreg-Corpus scheint von der Kombination des Graph-Kernels mit der Vereinigung der Merkmale zu profitieren. Es wurde für diese Kombination zusätzlich ein Schwellwertverfahren durchgeführt. Das Verfahren erreicht ein *F*-Maß von 63,98% und stellt damit die beste Methode speziell für das Genereg-Corpus dar.

Insgesamt sind die Resultate des Graph-Kerns und der Vereinigung von Merkmalen die besten. So ist es für die Extraktion von PPIs offenbar von Vorteil, größere Syntaxstrukturen zu betrachten. Die Verwendung von Syntaxanalysen scheint allgemein von Vorteil zu sein, wobei die Ergebnisse des Graph-Kerns auf dem LLL05-Material nicht so herausstechend sind, wie es für die Ergebnisse des Kerns auf AIMed oder Genereg

der Fall ist. Das könnte mit dem Link-Parser zusammen hängen, der zur syntaktischen Analyse des LLL05-Materials eingesetzt wurde. Das Parsingschema definiert sehr viel weniger Dependenztypen, als es bei anderen Dependenz-Schemata der Fall ist, und belässt häufiger Tokens völlig ohne Dependenzrelation. Zudem ist es offenbar hilfreich, die Syntaxinformationen möglichst weitreichend auszuschöpfen. Die Dependenzmerkmale aus Abschnitt 4.4 beziehen sich bereits aus die Syntaxanalyse, liegen in den Resultaten aber dennoch hinter den meisten anderen Verfahren zurück. Die Walk-Merkmale, die einen vollständigen Pfad des Syntaxbaums bzw -graphens verwenden, erreichen signifikant bessere Ergebnisse. Der Graph-Kern schließlich, der als einziges Verfahren in der Lage ist, volle, allgemeine Graphen zu berücksichtigen, funktioniert auf den größeren und aussagekräftigeren Corpora am besten. Es ist dabei nicht selbstverständlich anzunehmen, dass die Betrachtung des gesamten Dependenzgraphen stets von Vorteil ist. Die Extraktion von Ereignissen stellt eine verwandte Disziplin zur Relationsextraktion dar. Hierbei werden in der Regel ebenfalls zwei Parteien gesucht, die in einer Beziehung zueinander stehen. Der hier implementierte Graph-Kern wurde auch für die Ereignisextraktion verwendet, wobei eine Beschränkung auf den kürzesten Pfad die besten Resultate ergab[2].

Es ist außerdem festzuhalten, dass der Graph-Kern außerordentlich gut mit dem Genereg-Corpus zu arbeiten scheint. Da das Corpus nur PPI der Kategorie der Genregulationen enthält, bleibt zu vermuten, dass der Graph-Kern mit dieser Art von Relation besonders gut funktioniert.

Der Verteilungskern konnte erwartungsgemäß keine hohen Ergebnisse erzielen, was der Anzahl von unterschiedlichen Gen- und Proteinnamen zuzuschreiben ist. Im AIMed-Corpus konnten für 3407 von insgesamt 5834 vorhandenen potentiellen PPI keine Verteilungen der Argumente geschätzt werden. Für das Genereg-Corpus fehlten Verteilungen für 7196 von insgesamt 11514 potentiellen Relationen. Das lag einerseits daran, dass die entsprechenden Wörter nicht im TREC06-Material auffindbar waren. Weiterhin konnte es auch passieren, dass die Protein- oder Genbezeichnungen so selten vorkamen, dass sie nie in der Nähe eines Wortes aus dem Kookkurrenzvokabular beobachtet wurden. Zudem kommt es vor, dass die Argumente einer Relation aus mehreren Tokens bestehen. Da die Verteilungsschätzung sowie die Verteilungsmaße allerdings auf Tokens definiert sind, konnte dieser Tatsache keine Rechnung getragen werden. Diese Probleme treffen entsprechend den Local-Alignment-Kern, obwohl er nur teilweise auf der Verteilungsschätzung basiert und nicht nur Verteilungen bezüglich der Argumente berücksichtigt. Dass der Kern auf dem LLL05-Material gut funktioniert, hängt vermutlich mit der geringen Größe des Corpus zusammen. Da es verhältnismä-

[2]Dieses Ergebnis wurde durch persönliche Korrespondenz mitgeteilt.

ßig wenige Wörter enthält, ist es möglich, sich auf die vorhanden Wörter zu konzentrieren und sinnvolle Verteilungen zumindest für einige Wörter zu erlangen.

Die Ergebnisse von Merkmalen unter Verwendung des linearen Kerns lagen allgemein dicht mit den Werten zusammen, die durch Verwendung des RBF-Kerns erreicht wurden. Tatsächlich waren die Resultate des linearen Kerns oft besser. Es wurde bei den Experimenten beobachtet, dass stets sehr kleine Werte im Bereich 2^{-7} bis 2^{-6} für den RBF-Parameter γ selektiert wurden. Betrachtet man Gleichung 3.3.13, die den RBF-Kern beschreibt, so wird klar, dass so kleine Werte von γ den Absolutwert des Exponenten wachsen lassen. Durch das schnelle Wachstum der Exponentialfunktion kommt es zu einer extremen Steigung und folglich zu einer geringen Krümmung des Funktionsverlaufs. Der RBF-Kern stellt damit in diesem Fall eine Näherung an den linearen Kern dar. Dieses Verhalten ist ohne weiteres mit der Kern-Theorie vereinbar. Die Kernoperatoren werden verwendet, um die Eingabevektoren implizit in einen hochdimensionalen Raum abzubilden, wo sie linear separiert werden können. In Abschnitt 3.2 wurde erläutert, dass für die Merkmalgewinnung Wörter und andere Zeichenketten selbst als Indikatorfunktion verwendet werden. Als Beispiel sollen an dieser Stelle die Bag-of-Words-Merkmale dienen. Wird etwa das Wort *activates* beobachtet, so erhält es eine eigene Position i in den Merkmalsvektoren. Der Merkmalsvektor jeder potentiellen Interaktion erhält daraufhin den Wert Null, Eins oder Zwei an der Stelle i, je nachdem ob das Wort *activates* beobachtet wird und ob es sich zwischen den Argumenten des Interaktionskandidaten befindet. Da dieses Verfahren für jedes beobachtete Wort durchgeführt wird, ist der Merkmalsraum ohnehin von einer sehr hohen Dimension. Das macht den Einsatz weiterer Kernoperatoren in vielen Fällen unnötig oder sogar nicht wünschenswert. Diese Argumentation findet auch in Hsu u. a. (2008) Bestätigung.

Durch die Zusammenführung der hier betrachteten Methoden in ein zentrales Projekt können die Ansätze direkt miteinander verglichen werden. Es wird von jedem Corpus stets nur eine Version bzw. Interpretation genutzt und die Evaluationsmethodik ist stets die gleiche. Für den externen Vergleich werden Airola u. a. (2008) angeführt, die den Graph-Kern im Original implementiert haben. In diesem Aufsatz wurde auf dem AIMed-Corpus evaluiert. Dabei fand eine spezielle Aufteilung für die 10-fache Kreuzvalidierung Verwendung, die ein Resultat von 56% F-Maß hervor brachte. Die Aufteilung ist für diese Arbeit nicht verfügbar, allerdings wurde die Originalsoftware des Graph-Kerns heruntergeladen und mit dem gleichen AIMed-Corpus verwendet, das auch für die hier beschriebenen Auswertungen herangezogen wurde. Dabei wurde ein F-Maß von 52% erreicht. Das beste bekannte Ergebnis auf dem AIMed-Corpus, das auf dokumentenweiser Evaluation beruht, wurde von Miwa u. a. (2008) erreicht und beträgt 62% F-Maß. Es werden mehrere Kombinationen von Kernoperatoren, unter an-

derem ein Graph-Kern, und ein spezieller Algorithmus für das Schwellwertverfahren verwendet. Zudem wurde das Corpus mit unterschiedlichen, für diese Arbeit nicht verfügbaren Dependenzparsern analysiert. Das beste bisher bekannte Ergebnis auf dem Genereg-Corpus wurde mit dem Maximum-Entropy-Ansatz des JREX-Projekts erzielt und beträgt 48% *F*-Maß.

KAPITEL 9

Zusammenfassung und Ausblick

In dieser Arbeit wurden Ansätze zur automatischen Extraktion von Protein-Protein-Interaktionen in biomedizinischen Texten untersucht. Es wurde ein Lösungsansatz aus dem maschinellen Lernen gewählt, der aus Beispielmaterial ein Modell lernte, um PPIs zu erkennen und zu klassifizieren. Die gewählte Technik war die so genannte Supportvektormaschine, die eine klassentrennende Hyperebene im Merkmalsraum berechnete, der die zu klassifizierenden Objekte enthielt. Um auch nicht linear separierbare Datensituationen besser erfassen zu können, wurde der Kernel-Trick vorgestellt, durch den die Daten in einen hochdimensionalen Raum $\mathbb{H}$ abbgebildet wurden. Im Raum $\mathbb{H}$ konnten die Daten besser linear voneinander getrennt werden. Kernoperatoren $K(\cdot, \cdot)$ fanden Verwendung, um Skalarprodukte von Vektorpaaren zu berechnen. Die SVM war dann in der Lage, auf Grund dieser Skalarprodukte eine Hyperbene in $\mathbb{H}$ zu finden, die sich als nichtlineare Trennfunktion im Merkmalsraum manifestierte. Um Kernwerte zu erhalten, wurden einerseits Merkmalrepräsentationen von PPI verwendet, die in Zahlenvektoren $\boldsymbol{x}_i$ resultieren. Diese Vektoren wurden mit Kernoperatoren verwendet, die auf einer vektoriellen Repräsentation von Objekten arbeiteten. Andererseits wurden Techniken vorgestellt, die Kernwerte von Objekten berechneten, ohne sie explizit als Merkmalsvektor respräsentieren zu müssen.

Die verschiedenen Techniken wurden mit den Corpora AIMed, Genereg und LLL05 evaluiert. Die Verfahren wurden miteinander verglichen und nach Möglichkeit kombiniert, um bessere Ergebnisse zu erhalten. Das beste Verfahren für die PPI-Extraktion stellte der Graph-Kern dar, der als einzige Methode für den Vergleich vollständiger Graphen eingesetzt werden konnte. Dieser Kernoperator erhält Kernwerte durch den Vergleich zweier Graphen, die eine Dependenzanalyse jeweils eines Satzes darstellten. Der Graph-Kern erreichte nach einer Schwellwert-Nachprozessierung auf dem AIMed-Corpus 55,33% F-Maß und auf dem Genereg-Corpus 62,99%. Das Verfahren unterlag auf den LLL05-Daten einer Vereinigung von unterschiedlichen Merkmalmengen, die ein Resultat von 78,48% F-Maß hervor brachte. Insgesamt erwies sich die

Verwendung von Dependenzstrukturen für die Aufgabe der PPI-Extraktion als hilfreich. Durch Kombination der Kernwerte mehrerer Verfahren konnte auf dem Genereg-Corpus ein *F*-Maß von 63,98% erreicht werden. Davon abgesehen brachte die Kombination keine signifikanten Vorteile hervor.

Verfahren, die auf der Schätzung von Kookkurrenzwahrscheinlichkeiten beruhten, schnitten am schlechtesten ab. Die Gründe dafür sind zum Einen darin zu suchen, dass für viele Fachbegriffe der biomedizinischen Domäne auf Grund ihrer seltenen Verwendung keine zuverlässige Verteilung geschätzt werden konnte. Weiterhin waren die hier eingesetzten Verfahren zur Verteilungsschätzung verhältnismäßig einfach angelegt und verfügten eventuell nicht über die nötige Mächtigkeit, um repräsentative Wahrscheinlichkeitsverteilungen zu erstellen.

Insgesamt konnten durch Merkmale mit einfachen Mitteln gute Ergebnisse erzielt werden. Von drei Ansätzen, einen Kernoperator ohne vektorielle Repräsentationen zu verwenden, konnte hingegen nur der Graph-Kern gute Ergebnisse vorweisen. Gleichzeitig stellt diese Methode das beste Verfahren dar. Die Erstellung von Merkmalen scheint damit eine verhältnismäßig einfache Vorgehensweise zu sein, die potentiell sehr gute Ergebnisse hervorbringen kann. Durch die Schwierigkeit der Modellierung kann es aber ratsam sein, einen Kernoperator zu verwenden, der eine direkte Abbildung zweier Objekte auf einen Kernwert darstellt. Diese direkte Abbildung ist nicht an eine bestimmte Respräsentation der Objekte - wie Vektoren - gebunden, was die Realisierung einer Methode vereinfachen kann. Ob eine der beiden Vorgehensweisen - Merkmalextraktion oder direkte Abbildung - prinzipiell besser ist als die jeweils andere, kann an dieser Stelle nicht beantwortet werden und muss weiter untersucht werden.

Da das Prinzip der Verwendung von Kookkurrenzverteilung vielversprechend ist, könnte in zukünftigen Ansätzen versucht werden, die Güte der Verteilungsschätzung zu verbessern. Dies könnte einerseits durch Systeme geschehen, die relevante Passagen aus größeren Textsammlungen liefern, so genannte Passage Retrieval Systeme. Andererseits wurde hier mit Verteilungen und Ähnlichkeitsmaßen auf Verteilungen gearbeitet, die auf der Basis von Tokens operierten. Da jedoch die Argumente von PPI mehrere Tokens umfassen können, wäre eine Entwicklung von Methoden hilfreich, die auch solche größeren Strukturen geeignet einbeziehen würden.

Repräsentative Verteilungen könnten als zusätzliche Information in den Graph-Kern einfließen. Der Kern vergibt Gewichte an Graph-Kanten, die einem einfachen Gewichtungsschema folgen. Wäre eine zuverlässige Kookkurrenzinformation zweier Wörter bekannt, könnte sie in die Kantengewichtung mit einbezogen werden.

Das beste Ergebnis auf dem AIMed-Corpus wurde von Miwa u. a. (2008) berich-

tet, wobei auch dort ein Graph-Kern verwendet wird. Dabei wurde einige Aufmerksamkeit auf die syntaktische Analyse des Datenmaterials gelegt. Es liegt nahe anzunehmen, dass der Graph-Kern zusammen mit bestimmten Dependenzanalysen besonders gute Resultate erzielt. Solche Verbindungen sollten in Zukunft weiter untersucht werden. Zudem wird in eben genanntem Aufsatz einer Kombination mehrer Ansätze beschrieben, wie sie auch in der der vorliegenden Arbeit untersucht wurde. Die Resultate von Miwa u. a. (2008) wurden durch die Kombination verbessert, so dass nach weiteren Methoden gesucht werden sollte, die für eine Kombination gewinnbringend sein könnten.

Glossar

Adjazenzmatrix Die Adjazenzmatrix eines gewichteten Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ ist eine Matrix $A \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$. Die Elemente a_{ij} der Adjazenzmatrix A sind gegeben durch das Kantengewicht der Kante von $v_i \in \mathcal{V}$ nach $v_j \in \mathcal{V}$ oder 0, falls keine Kante existiert, 44

Chunking Chunking bezeichnet das automatische Zusammenfassen einer zusammenhängenden Wortgruppe zu einer Phrase, 9, 62

Corpus Ein Corpus ist eine Sammlung von Texten oder Äußerungen einer, zumeist natürlichen, Sprache, 1

Dependenzgrammatik Eine Dependenzgrammatik strukturiert einen Satz, indem seine Tokens durch Abhängigkeitsrelationen miteinander in Beziehung gesetzt werden, 9

Development Set Siehe Entwicklungssatz, 54

Entwicklungssatz Teil des Datenmaterials in Anwendungen des maschinellen Lernens, der für die Parameterschätzung verwendet wird, 54

F-Maß Das F-Maß ist ein Maß zur Auswertung einer IE-Methode. Es ergibt sich aus dem harmonischen Mittel von Precision und Recall, 49

False Negative Ein False Negative ist bei der Klassifikatorauswertung ein Objekt, das fälschlicherweise als keiner Klasse von Interesse zugehörig erkannt wurde, 49

False Positive Ein False Positive ist bei der Klassifikatorauswertung ein Objekt, das zwar erkannt, aber in die falsche Klasse eingeteilt wurde, 49

Feature Siehe Merkmal, 12

Feature Subset Selection Die Feature Subset Selection bezeichnet allgemein das Auswählen von Merkmalfunktionen $f_{i_1}, f_{i_2} \ldots, f_{i_n}$ aus einer Gesamtmenge von Merkmalen $f_1, \ldots, f_m$. Der Zweck der Auswahl ist zumeist die Suche nach einem (lokalen) Optimum in der Erkennungsrate eines Algorithmus' des maschinellen Lernens, 30

Gittersuche Bei der Gittersuche wird der Parameterraum eines Verfahrens in ein logarithmisches Gitter aufgeteilt. Die Gitterpunkte in einem Bereich von Interesse werden auf ihre Auswirkung auf das Klassifikationsergebnis hin untersucht, 54, 64

Gram-Matrix Die Gram-Matrix G enthält als Einträge die paarweisen Skalarprodukte der Vektoren einer Menge $\mathfrak{X} = \{ \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \}$. G ist also von der Gestalt $G = \{ g_{i,j} = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \}_{i,j=1}^{m,m}$, 25

Grid Search Siehe Gittersuche, 54

Kernel-Trick Der Kernel-Trick bezeichnet die Berechnung von Skalarprodukten $\langle \phi\boldsymbol{x}, \phi\boldsymbol{y} \rangle$ von nichtlinear expandierten Vektoren $\phi\boldsymbol{x}, \phi\boldsymbol{y}$ mit Hilfe eines Kernoperators $K(\cdot, \cdot)$. Die nichtlineare Abbildung ϕ muss bei diesem Verfahren nicht explizit bekannt sein, 21, 87

Kernoperator Ein Kernoperator berechnet das Skalarprodukt der (nichtlinearen) Einbettung zweier Vektoren in einen höherdimensionalen Raum, 1, 21

Koordination Die Verbindung zweier gleichwertiger Wörter oder sogar Sätze durch eine Konjunktion, 5

Kreuzvalidierung Die Kreuzvalidierung ist ein Verfahren zur Auswertung eines Klassifikators. Das vorhandene Datenmaterial wird für eine n-fache Kreuzvalidierung in n gleich große, überlappungsfreie Teile partitioniert. Anschließend werden je $n - 1$ Partitionen für das Training verwendet, die überbleibende Partition dient zum Test. Dieses Verfahren wird wiederholt, bis jede Partition einmal als Testmaterial fungiert hat, 50

Label Siehe Tag, 8

Leave-One-Out Leave-One-Out-Validierung bezeichnet eine Kreuzvalidierung mit der maximalen Anzahl von Partitionen, 50

Levenshtein-Abstand Der Levenstheinabstand transformiert durch die Operationen Einfügung, Löschung und Austausch von Zeichen eine Zeichenkette x in eine Zielzeichenkette y, wobei jede Operation spezielle Kosten verursacht. Der Abstand der beiden Zeichenketten ist durch die günstigste Folge von Operationen definiert und stellt damit ein globales Maß für den Abstand zweier Zeichenketten zueinander dar, 40

Local Alignment Ein local Alignment zwischen zwei Zeichenketten x und y ist eine hohe Übereinstimmung oder Ähnlichkeit zweier Teilketten x' und y' aus x bzw. y, 44

Maschine Learning Siehe maschinelles Lernen, 30

Maschinelles Lernen Oberbegriff für Verfahren, die aus Beispieldaten allgemeine Regeln lernen, um unbekannte Daten einer von K vordefinierten Kategorien zuzuweisen, 1, 30, 49, 51, 61

Maximum Entropy in statistisches Klassifikationsverfahren, das dem Grundprinzip von Occams Rasiermesser folgt ('die einfachste Erklärung ist stets die beste'), 61, 85

Maximum Likelihood Maximum Likelihood ist ein Verfahren zur Schätzung von parametrisierten Wahrscheinlichkeitsverteilungen. Gegeben eine Stichprobe ermittelt das Verfahren denjenigen Parametersatz, der die Wahrscheinlichkeit der vorliegenden Beobachtung - der Stichprobe - maximiert. Im Falle diskreter Wahrscheinlichkeitsverteilungen ergeben sich dadurch die relativen Wahrscheinlichkeiten der gemachten Beobachtungen, 69

Merkmal Als Merkmale werden in der Musteranalyse die Ausprägungen der einzelnen Attribute eines Objekts aus dem betrachteten Problemkreis bezeichnet, 12

Named Entity Als Named Entity wird hauptsächlich alles bezeichnet, das durch einen Eigennamen referenziert werden kann, aber auch weitere Informationen wie Gewichts- oder Datumsangaben, die von praktischer Relevanz sind, 9

Named Entity Recognition Die Named Entity Recognition bezeichnet die Aufgabe des Auffindens von benannten Entitäten bzw. Named Entitys in einem gegebenen Text, 8, 9, 62

Natural Language Processing Natural Language Processing beschäftigt sich mit der automatischen Verarbeitung von natürlicher Sprache, 12

Parsing Parsing bezeichnet die Erstellung einer strukturellen Analyse einer Wortfolge, zumeist eines Satzes, 9, 62

Passage Retrieval System Ein Passage Retrieval System extrahiert automatisch relevante Passagen bezüglich eines Eingabewortes aus einer Textsammlung, 67

Phrasenstrukturgrammatik Eine Phrasenstrukturgrammatik erlaubt die Ableitung von Sätzen mit einer Konstituentenstruktur, 9

PoS-Tagging Siehe Wortartenerkennung, 8

Precision Precision ist ein Maß zur Auswertung einer IE-Methode. Sie sagt aus, wie viele von den insgesamt als positiv erkannten Objekten richtig klassifiziert wurden, 49

Recall Recall ist ein Maß zur Auswertung einer IE-Methode. Sie sagt aus, wie viele von den insgesamt auffindbaren Objekten erkannt wurden, 49

Satzerkennung Satzerkennung meint die Aufgabe, automatisch Sätze in einem natürlichsprachigen Text zu identifizieren, 8, 62

Schwellwert Bei einem Schwellwertverfahren wird die Zugehörigkeit eines Objekts x potentiell nach der Klassifikation in die Klasse κ nachträglich geändert. Die Änderung richtet sich danach, ob die Wahrscheinlichkeit - oder ein anderes Maß - für eine andere Klasse $\lambda \neq \kappa$ einen bestimmten Schwellwert T überschreitet, 16

Shortest Path Hypothesis Die Shortes Path Hypothesis - Hypothese des kürzesten Pfades - nimmt an, dass die wichtigsten lexikalisch-semantischen Informationen, um eine PPI in einem Satz anzuzeigen, im kürzesten Dependenzpfad zwischen den potentiellen Argumenten enthalten sind, 34, 40, 43

Smith-Waterman-Maß Das Smith-Waterman-Maß ähnelt prinzipiell dem Levenstein-Abstand. Jedoch berechnet es keinen globalen Abstand sondern eine maximale, lokale Ähnlichkeit zwischen zwei Zeichenketten, 40

Socket Im Zusammenhang mit der TCP/IP-Protokollsuite werden eine IP Adresse eines Computers und der TCP-Port einer Anwendung als Socket zusammengefasst, 65

Spektralradius Der Spektralradius einer Matrix A ist gegeben durch den Betrag des betragsgrößten Eigenwerts von A, 47

Stoppwortliste Eine Stoppwortliste enthält oft vorkommende Wörter ohne spezifische Semantik. Diese Wörter werden durch Verwendung der Stoppwortliste aus einer Anwendung herausgefiltert, da sie die Resultate verrauschen würden. Typischerweise enthalten sind die Artikel einer Sprache, ihre Funktionswörter und weitere Wörter, die generell extrem häufig auftauchen, 69

Supportvektor Supportvektoren sind diejenigen Vektoren des Trainingsmaterials, die genau auf dem Rand der klassentrennenden Hyperebene liegen. Sie legen damit Position und Lage der Ebene fest, 5, 20, 61

Supportvektormaschine Ein Verfahren im Machine Learning, bei dem ein unbekanntes Objekt durch die Partitionierung des Objektraums in zwei Hälften bzw. Seiten klassifiziert wird, 1, 13

Tag Eine Klassenzugehörigkeitsmarkierung für ein textuelles Objekt wie beispielsweise ein PoS-Tag für ein Token, 8

Terminale Terminale bezeichnen nicht weiter ableitbare Symbole einer PSG. Eine vollständige Ableitung einer PSG besteht ausschließlich aus Terminalen, 9

Tokenisierung Tokenisierung ist die automatische Erkennung von Tokens in einem gegebenen Text, 8, 12, 62

True Positive Ein True Positive ist bei der Klassifikatorauswertung ein Objekt, das der richtigen Klasse zugewiesen wurde, 49

Wortartenerkennung Bei der Wortartenerkennung wird jedem Token in einem Text seine Wortart zugewiesen, 8, 62

Zusammenhangskomponente Eine Zusammenhangskomponente eines Graphen $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ ist eine Menge von Knoten aus $\mathcal{V}$, die durch Kanten aus $\mathcal{E}$ miteinander verbunden sind. Ein Graph besitzt mehrere Zusammenhangskomponenten, wenn es mindestens zwei Knoten $v_i, v_j \in \mathcal{V}$ gibt, so dass kein Pfad zwischen v_i und v_j existiert, 44

Literatur

Airola, Antti, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter und Tapio Salakoski (2008). „A Graph Kernel for Protein-Protein Interaction Extraction“. In: *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing (BioNLP'08)*. Association for Computational Linguistics, S. 1–9.

Bunescu, Razvan C. und Raymond J. Mooney (2005). „A shortest path dependency kernel for relation extraction“. In: *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada: Association for Computational Linguistics, S. 724–731.

Bunescu, Razvan, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun K. Ramani und Yuk W. Wong (2005). „Comparative experiments on learning information extractors for proteins and their interactions“. In: *Artificial Intelligence in Medicine* 33.2, S. 139–155.

Buyko, Ekaterina, Elena Beisswanger und Udo Hahn (2008). „Testing Different ACE-Style Feature Sets for the Extraction of Gene Regulation Relations from MEDLINE Abstracts“. In: *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM 2008), Turku, Finland*. Hg. von Tapio Salakoski, Dietrich R. Schuhmann und Sampo Pyysalo. Turku Centre for Computer Science (TUCS), S. 21–28.

Chang, Chih-Chung und Chih-Jen Lin (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Charniak, Eugene und Matthew Lease (2005). „Parsing biomedical literature“. In: *In Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP-05)*, S. 58–69.

Doddington, George, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel und Ralph M. Weischedel (2004). „The Automatic Content Extraction (ACE) Program. Tasks, Data, and Evaluation“. In: *LREC 2004 – Proceedings of the 4th International Conference on Language Resources and Evaluation*. Lisbon, Portugal.

Fundel, Katrin, Robert Küffner und Ralf Zimmer (2006). „RelEx—Relation extraction using dependency parse trees“. In: *Bioinformatics* 23.3, S. 365–371.

Grefenstette, G. und P. Tapanainen (1994). *What is a word, what is a sentence? problems of tokenization*.

GuoDong, Zhou, Su Jian, Zhang Jie und Zhang Min (2005). „Exploring various knowledge in relation extraction“. In: *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Ann Arbor, Michigan: Association for Computational Linguistics, S. 427–434.

Hahn, Udo und Joachim Wermter (2006). *Text Mining for Biology And Biomedicine*. Hg. von Sophia Ananiadou und John Mcnaught. Artech House Publishers.

Hersh, William, Phoebe Roberts Aaron M. Cohen und Hari Krishna Rekapalli (2006). „TREC 2006 Genomics Track Overview“. In: *Proceedings of the TREC Genomics Track 2006*. Oregon Health & Science University, USA.

Hsu, Chih-Wei, Chih-Chung Chang und Chih-Jen Lin (2008). *A Practical Guide to Support Vector Classification*. Department of Computer Science, National Taiwan University, Taipei 106, Taiwan. 02. Okt. 2008.

Jurafsky, Daniel und James H. Martin (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall.

Katrenko, Sophia und Pieter Adriaans (2007). „Learning Relations from Biomedical Corpora Using Dependency Trees“. In: *Knowledge Discovery and Emergent Complexity in Bioinformatics*, S. 61–80.

— (2008). „A Local Alignment Kernel in the Context of NLP“. In: *Proceedings of the 22nd International Conference on Computational Linguistics*. Manchester, UK, S. 417–242.

Kim, Seonho, Juntae Yoon und Jihoon Yang (2008). „Kernel approaches for genic interaction extraction“. In: *Bioinformatics* 24.1, S. 118–126. ISSN: 1367-4803.

Marneffe, M., B. Maccartney und C. Manning (2006). „Generating Typed Dependency Parses from Phrase Structure Parses“. In: *Proceedings of LREC-06 (The fifth international conference on Language Resources and Evaluation)*. Magazzini del Cotone Conference Center, Genua, Italien, S. 449–454.

Mcdonald, Ryan, Kevin Lerman und Fernando Pereira (2006). „Multilingual dependency analysis with a two-stage discriminative parser“. In: *In Proceedings of the Conference on Computational Natural Language Learning (CoNLL*. New York, USA, S. 216–220.

Miwa, Makoto, Rune Sætre, Yusuke Miyao, Tomoko Ohta und Jun'ichi Tsujii (2008). „Combining Multiple Layers of Syntactic Information for Protein-Protein Interaction Extraction“. In: *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM 2008), Turku, Finland*. Hg. von Tapio Salakoski, Dietrich

R. Schuhmann und Sampo Pyysalo. Turku Centre for Computer Science (TUCS), S. 101–108.

MUC-7 Named Entity Task Definition (1997).
http://www.itl.nist.gov/iad/894.02/related_projects/muc/proceedings/ne_task.html. 17. Sep. 1997.

Nédellec, Claire (2005). „Learning Language in Logic Genic Interaction Extraction Challenge“. In: *Proceedings of the Learning Language in Logic workshop*. Universität Bonn, Deutschland.

Niemann, Heinrich (1990). *Pattern Analysis and Understanding*. Berlin: Springer Verlag.

Pyysalo, Sampo, Antti Airola, Juho Heimonen, Jari Bjorne, Filip Ginter und Tapio Salakoski (2008). „Comparative analysis of five protein-protein interaction corpora“. In: *BMC Bioinformatics* 9.

Saigo, Hiroto, Jean-Philippe Vert, Nobuhisa Ueda und Tatsuya Akutsu (2004). „Protein homology detection using string alignment kernels“. In: *Bioinformatics* 20.11, S. 1682–1689.

Schölkopf, Bernhard und Alexander J. Smola (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press.

Schukat-Talamazzini, Ernst Günter (2005). *Mustererkennung: Vorlesung im Sommersemester (Vorlesungsskriptum*. Universität Jena, Deutschland.

Séaghdha, Diarmuid Ó und Ann Copestake (2008). „Semantic classification with distributional kernels“. In: *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*. Manchester, UK.

Sleator, C Daniel und Davy Temperley (1993). „Parsing English with a link grammar“. In: *In Proceedings of the 3rd International Workshop on Parsing Technologies*. Universität Tilburg, Die Niederlanden.

Smith, T. F. und M. S. Waterman (1981). „Identification of common molecular subsequences“. In: *Journal of Molecular Biology* 147.1 (März 1981), S. 195–197.

PUBMED (2008).
http://www.nlm.nih.gov/bsd/index_stats_comp.html. 01. Okt. 2008.

UniProt (2009).
http://www.uniprot.org/.

WirtschaftsWoche (2008).
http://www.wiwo.de/unternehmer-maerkte/vodafone-greift-bei-der-tochter-arcor-durch-261637/. 12. Jan. 2008.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

Analysis Engine (AE) Prozessierungsmodul im UIMA-Framework, 62, 66

Common Analysis System (CAS) Ein einheitliches Repräsentationssystem für Daten, das vom UIMA-Framework definiert und genutzt wird, 62

Database of Interacting Proteins (DIP) Datenbank von medizinischen Texten, die PPI beschreiben, 57

False Negative (FN) Ein False Negative ist bei der Klassifikatorauswertung ein Objekt, das fälschlicherweise als keiner Klasse von Interesse zugehörig erkannt wurde, 49

False Positive (FP) Ein False Positive ist bei der Klassifikatorauswertung ein Objekt, das zwar erkannt, aber in die falsche Klasse eingeteilt wurde, 49, 51

Informationsextraktion (IE) Informationsextraktion bezeichnet die Aufgabe, strukturierte Informationen aus einem unstrukturierten Text zu extrahieren, 3, 91, 94

Least Common Subsumer (LCS) Der LCS bezeichnet den kleinsten ('niedrigsten') gemeinsamen Vorgänger von zwei Knoten in einem Baum, 33, 34

Maximum Entropy (ME) Ein statistisches Klassifikationsverfahren, das dem Grundprinzip von Occams Rasiermesser folgt ('die einfachste Erklärung ist stets die beste'), 61

Machine Learning (ML) Oberbegriff für Verfahren, die aus Beispieldaten allgemeine Regeln lernen, um unbekannte Daten einer von K vordefinierten Kategorien zuzuweisen, 30

Message Understanding Conference (MUC) Eine Konferenzserie, die sich mit dem Thema der Informationsextraktion beschäftigt, 9

Named Entity (NE) Als Named Entity wird hauptsächlich alles bezeichnet, das durch einen Eigennamen referenziert werden kann, aber auch weitere Informationen wie Gewichts- oder Datumsangaben, die von praktischer Relevanz sind, 3, 5, 7, 9, 32, 51, 57, 58

Named Entity Recognition (NER) Die Named Entity Recognition bezeichnet die Aufgabe des Auffindens von benannten Entitäten bzw. Named Entitys in einem gegebenen Text, 5, 62, 65

Natural Language Processing (NLP) Natural Language Processing beschäftigt sich mit der automatischen Verarbeitung von natürlicher Sprache, 12

Part-of-Speech (PoS) Part-of-Speech ist die englische Bezeichnung für Wortart, 8, 65

Protein-Protein-Interaktion (PPI) Als PPI werden semantische Relationen zwischen Proteinen aber auch Genen bezeichnet, die in die allgemeine Klasse der Interaktion fallen, 1, 2, 4, 10, 12, 29–32, 34, 39, 45, 47, 49, 51, 52, 54, 57, 58, 61, 64–67, 77, 83, 84, 87, 88, 94, 105

Phrasenstrukturgrammatik (PSG) Eine Phrasenstrukturgrammatik erlaubt die Ableitung von Sätzen mit einer Konstituentenstruktur, 9, 95

Supportvektormaschine (SVM) Ein Verfahren im Machine Learning, bei dem ein unbekanntes Objekt durch die Partitionierung des Objektraums in zwei Hälfte bzw. Seite klassifiziert wird, 13, 14, 16–18, 20, 24–26, 29, 37, 38, 49, 53–55, 61–64, 75, 87

True Positive (TP) Ein True Positive ist bei der Klassifikatorauswertung ein Objekt, das der richtigen Klasse zugewiesen wurde, 49

Unstructured Information Management Architecture (UIMA) UIMA stellt ein Framework für die Verarbeitung von unstrukturierten Daten zum Zwecke der strukturierten Darstellung dieser Daten, 62, 66

ANHANG A

Merkmallisten

A.1 Wort-Merkmale

In Tabelle A.1 werden die Wort-Merkmale aus Abschnitt 4.6 aufgelistet. E1 und E2 bezeichnen dabei den ersten bzw. zweiten Argumentkandidaten einer potentiellen Relation. Die Merkmale beziehen sich auf einen Satz, in dem E1 und E2 gemeinsam vorkommen.

A.2 Merkmale bezüglich Basisphrasen-Chunks

Tabelle A.2 liefert eine Übersicht über die Merkmale betreffend Basisphrasen-Chunks aus Abschnitt 4.6. Wie bei den Wort-Merkmalen seien die beiden betrachteten Argumentkandidaten mit E1 und E2 bezeichnet. Alle Merkmale beziehen sich wieder auf einen Satz, der E1 sowie E2 enthält.

A.3 Parse-Merkmale

Die Parse-Merkmale aus Tabelle A.3 beziehen sich auf die Konstituenten an sich und auf den Konstituenten-Pfad, der die potentiellen Argumente E1 und E2 miteinander verbindet. Es handelt sich hier um Merkmale, auf die in Abschnitt 4.6 referenziert wird.

Merkmal	Beschreibung
WBNULL	ob kein Wort zwischen E1 und E2 vorkommt
WBFL	das Wort zwischen E1 und E2, falls genau ein Wort zwischen den Entitäten steht
WBF	das erste Wort zwischen E1 und E2, falls mindestens zwei Wörter zwischen den Entitäten stehen
WBL	das letzte Wort zwischen E1 und E2, falls mindestens zwei Wörter zwischen den Entitäten stehen
WBO	weitere Wörter aus dem ersten und letzten Wort zwischen E1 und E2, falls mindestens drei Wörter zwischen den Entitäten stehen
BM1F	das erste Wort vor E1
BM1L	das zweite Wort vor E1
AM2F	das erste Wort nach E2
AM2L	das zweite Wort nach E2

Tabelle A.1: Wort-Merkmale für die Relationsextraktion.

Merkmal	Beschreibung
CPHBNULL	ob keine Phrase zwischen E1 und E2 vorkommt
CPHBFL	der Kopf der Phrase zwischen E1 und E2, falls genau eine Phrase zwischen den Entitäten steht
CPHBF	der Kopf der ersten Phrase zwischen E1 und E2, falls mindestens zwei Phrasen zwischen den Entitäten stehen
CPHBL	der Kopf der letzten Phrase zwischen E1 und E2, falls mindestens zwei Phrasen zwischen den Entitäten stehen
CPHBO	die Köpfe weiterer Phrasen außer der ersten und letzten Phrase zwischen E1 und E2, falls mindestens drei Phrasen zwischen den Entitäten stehen
CPHBM1F	Kopf der ersten Phrase vor E1
CPHBM1L	Kopf der zweiten Phrase vor E1
CPHAM2F	Kopf der ersten Phrase nach E2
CPHAM2F	Kopf der zweiten Phrase nach E2
CPP	der Pfad der Bezeichnungen der Phrasentypen zwischen E1 und E2
CPPH	der Pfad aus CPP, elementweise erweitert durch Köpfe der entsprechenden Phrasen; dieses Merkmal wird nur generiert, falls höchstens zwei Phrasen zwischen E1 und E2 stehen

Tabelle A.2: Merkmale bezüglich Basisphrasen-Chunks für die Relationsextraktion.

Merkmal	Beschreibung
ET12SameNP	Kombination der Entitäten-Kategorien von E1 und E2 sowie einer booleschen Markierung, ob E1 und E2 in der gleichen Nominalphrase enthalten sind
ET12SamePP	Kombination der Entitäten-Kategorien von E1 und E2 sowie einer booleschen Markierung, ob E1 und E2 in der gleichen Präpositionalphrase enthalten sind
ET12SameVP	Kombination der Entitäten-Kategorien von E1 und E2 sowie einer booleschen Markierung, ob E1 und E2 in der gleichen Verbalphrase enthalten sind
PTP	Pfad der Konstituentenbezeichnungen zwischen E1 und E2 im Strukturbaum, wobei Duplikate aus dem Pfad entfernt werden
PTPH	PTP mit der Erweiterung der Konstituentenköpfe für die Elemente des Pfads

Tabelle A.3: Parse-Merkmale für die Relationsextraktion.

Persönliche Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

27. März 2009

Erik Fäßler