Bestimmung von Text-Patterns für die Informationsextraktion

Diplomarbeit

Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

ingereicht von: Conrad Plake
Betreuer: Prof. Dr. Ulf Leser, Dipl. Inf. Jörg Hakenberg

Berlin, 5. April 2005
Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Diplomarbeit selbstständig und nur unter Zuhilfenahme der angegebenen Quellen erstellt habe.

Berlin, den 5. April 2005  Conrad Plake
Inhaltsverzeichnis

1 Einleitung 1

2 Grundlagen 7

3 Methoden 11
   3.1 Modelle ........................................... 13
      3.1.1 Sequenzalignment .............................. 14
      3.1.2 Reguläre Ausdrücke ............................ 16
      3.1.3 Hidden Markov Modelle ......................... 17
      3.1.4 Vector Space Modelle ............................ 19
   3.2 Lernverfahren ....................................... 19
      3.2.1 Patterngenerierung .............................. 20
      3.2.2 Stochastische Optimierung ..................... 22
      3.2.3 Genetische Algorithmen ......................... 22

4 Software 25
   4.1 Tools ............................................... 25
   4.2 Eine Textmining-Pipeline ............................ 27

5 Anwendung: Protein-Protein-Interaktionen 29
   5.1 Vorverarbeitung ..................................... 32
   5.2 Pattern-Alignment .................................... 34
   5.3 Endliche Zustandsautomaten ......................... 36
   5.4 Implementation ...................................... 39
   5.5 Evaluation .......................................... 47
       5.5.1 Der BioCreative Corpus ........................ 47
5.5.2 Der IEPA-Korpus ........................................ 51
5.5.3 Die DIP ................................................. 53
5.6 Fehleranalyse ............................................. 57
  5.6.1 Pattern-Alignment .................................... 58
  5.6.2 Zustandsautomaten ................................... 62
5.7 Zeitanalyse ................................................. 68

6 Zusammenfassung 71

7 Ausblick 75

A Sequenz-Diagramme 77
Kapitel 1

Einleitung


abgeleitete Informationen korrekt extrahieren zu können. Die Entwicklung von NER-Tools, besonders in einem so schnell wachsenden Forschungsgebiet wie den Life Sciences, ist daher ein aktuelles Forschungsthema.


Eine interessante Anwendung der Informationsextraktion ist die Erkennung von Protein-Protein-Interaktionen (PPI) in online verfügbaren, wissenschaftlichen Veröffentlichungen. Viele PPI sind ausschließlich in der Literatur beschrieben und müssen zunächst in strukturierte Daten überführt werden, um sie in einer Datenbank zu speichern und damit für die rechnergestützte Analyse nutzbar zu machen. Die Extraktion von PPI aus wissenschaftlichen Texten ist ein aktuelles Forschungsthema, zu dem in den vergangenen Jahren viele Methoden publiziert wurden.

Proteinbezeichner zurückzuführen und mit Interaktionsverben zu assozieren [Sekimizu et al., 1998]. Extraktionssysteme mit kontextfreien Grammatiken (Full Sentence Parser) müssen sehr viele syntaktische Ableitungen interpretieren, was auch viel Zeit in Anspruch nimmt. Anhand einer Ontologie kann die Menge an semantischen Ableitungen für einen Satz aber weiter eingeschränkt werden [Daraselia et al., 2004]. Solche Systeme erzielen eine gute Genauigkeit, jedoch ist ihr Durchsatz an Texten aufgrund des Rechenaufwandes wesentlich geringer. Huang et al. alignieren annotierte Sätze und leiten aus den übereinstimmenden POS-Annotationen Patterns ab, wenn zusätzliche Filterregeln erfüllt sind. Anhand der generierten Patterns werden Alignments zu neuen Sätzen berechnet und die passenden Textstellen extrahiert [Huang et al., 2004].

Diese Arbeit beginnt mit der Definition grundlegender Begriffe, die nötig sind um Texte zu formulieren und eine Informationsextraktion in Form darin benannter Entitäten und deren Relationen zu ermöglichen, gefolgt von einem Überblick einiger Methoden, die für eine Bestimmung der Relationen in einem Text geeignet sind. Mit der exemplarischen Anwendung der Erkennung von Interaktionen zwischen Proteinen werden zwei Extraktionsverfahren, Pattern-Alignment (Kapitel 5.2) und Zustandsautomaten (Kapitel 5.3), genauer untersucht und verglichen.


In dem zweiten Verfahren sind Patterns allgemein als endliche Zustandsautomaten repräsentiert, die nach Muster der generierten Patterns aus dem ersten Verfahren konstruiert sind oder von Grund auf neu gelernt werden, wofür ein Genetischer Algorithmus (GA) gute Ergebnisse erzielt. Nach dem Parsen werden die Entitäten in den passenden Textabschnitten indiziert und die relevanten Relationen anhand der zum Pattern bekannten Indextupel bestimmt.

Literaturquellen nachzuweisen. Auf der beiliegenden CD ist das entwickelte 
Java-Framework zusammen mit der Beispielanwendung, der Dokumentation 
und den drei Textkorpora enthalten.

In einer abschließenden Fehleranalyse werden die typischen Fehlerquellen 
und Probleme der untersuchten Verfahren genauer bestimmt und diskutiert. 
Die vorgestellten Lösungen und Ergebnisse der Arbeit werden noch einmal 
zusammengefasst und schließlich ein Ausblick auf weitere Arbeitsansätze ge-
geben.
Kapitel 2

Grundlagen

Um Informationen aus einem Text zu extrahieren, muss zunächst definiert werden, was unter einem Text zu verstehen ist und was die Informationen sind, die aus ihm extrahiert werden sollen. Ein Text besteht aus einer Folge von Wörtern und Satzzeichen, die in der Literatur verwendet werden und die hier alle unter dem Begriff *Token* zusammengefasst sind. Für die textbasierte Informationsextraktion ist vor allem die Bedeutung der einzelnen Tokens entscheidend. Bei der Annotation eines Textes wird jedes Token mit einem *Tag* verknüpft und so einer semantischen Bedeutung zugeordnet. Ein Tag ist eine beliebige Zeichenkette und kann in sich auch mehrere Bedeutungen tragen. Ein Tupel aus Token und Tag wird im Folgenden ein annotiertes Token, oder kurz Wort, genannt.

**Definition 2.1** Sei $TOK$ die Menge aller Zeichenketten, dann bezeichnet $W \subseteq (TOK \times TOK)$ eine Menge von Wörtern.

Ein Text ist eine Folge von Wörtern, muss keinen bestimmten Anfang oder ein bestimmtes Ende haben, sondern kann als eine beliebige Textpassage oder Phrase verstanden werden.

**Definition 2.2** Sei $W$ die Menge aller Wörter. Ein Text ist gegeben als eine Folge von Wörtern $t \in W^*$. Die Länge von $t$ wird mit $|t| = n$ bezeichnet.

Aus einem annotierten Text können nun Informationen, worunter in erster Linie die darin enthaltenen benannten Entitäten zu verstehen sind, extrahiert
werden. Eine textuelle Entität ist eine bestimmte Folge von Wörtern in einem Text und daher selbst wieder ein Text.

**Definition 2.3** Sei \( t = [w_0, w_1, \ldots, w_{n-1}] \) ein Text der Länge \( n \). Eine Entität in \( t \) ist eine Wortfolge \( (w_i, w_{i+1}, \ldots, w_{i+j}) \) mit \( i, j \geq 0 \land i + j \leq n - 1 \).

Für die Anwendung im Kapitel 5 werden spezielle Entitäten (Proteine) definiert, deren Wörter sich durch ein gemeinsames Tag (‘PTN’) auszeichnen. Für ein tieferes Verständnis von einem Text sind die Beziehungen der enthaltenen Entitäten zueinander bedeutend und werden deshalb nach Möglichkeit als weitere Information extrahiert. Eine Relation zwischen Entitäten ist wie folgt definiert.

**Definition 2.4** Sei \( E \) eine Menge von Entitäten. Eine Relation ist gegeben als ein Tupel \( r \in E^m, m \geq 2 \).

In einem Text wird nicht selten eine Entität mehrfach erwähnt und manchmal auch auf unterschiedliche Weise, so dass es erwünscht ist, diese Nominierungen aufeinander zurückzuführen. Hierfür wird eine Normalisierung durchgeführt, bei der neu extrahierte Entitäten ihren zugehörigen Synonymklassen zugeordnet werden.

**Definition 2.5** Seien \( E, S \) zwei Mengen von Entitäten, wobei letztere aus bekannten Synonymen besteht, \( \mathcal{P}(S) \) die Potenzmenge von \( S \) und \( S \subseteq \mathcal{P}(S) \) eine Teilmenge davon, deren Elemente Synonymklassen genannt werden. Eine Normalisierung ist eine Funktion, die eine Entität anhand einer Menge von Synonymklassen auf eine Synonymklasse abbildet:

\[
\text{norm} : E \times \mathcal{P}(S) \longmapsto S. \tag{2.1}
\]

Für eine konkrete Realisierung der Funktion \( \text{norm} \) werden beispielsweise die Entitäten mit den Synonymen paarweise verglichen und der Klasse des ähnlichsten Synonyms zugeordnet. Für die Berechnung der Ähnlichkeit zwischen zwei Entitäten wird die folgende Funktion eingeführt:

\[
\text{sim} : E \times E \longmapsto \mathbb{R}. \tag{2.2}
\]
Kapitel 3
Methoden

Für eine textbasierte Informationsextraktion kommen verschiedene Methoden in Frage, von denen sich einige auf das Parsen eines Textes mit Patterns oder Grammatiken, andere z.B. auf die Berechnung von Alignments zwischen Patterns und Sätzen zurückführen lassen. In dem folgenden Kapitel werden alle in dieser Arbeit angewendeten Modelle und Lernverfahren beschrieben.

Die zur Informationsextraktion verwendeten Text-Patterns sollen aus einem Trainingskorpus automatisch gewonnen werden, weshalb für ihre Bestimmung im besonderen Methoden anzuwenden sind, die sich für ein Lernen anhand von Trainingsbeispielen (Supervised-Learning) eignen. Des Weiteren wird davon ausgegangen, dass ein Text in annotierter Form, entsprechend der Definition 2.2, repräsentiert ist. Dies setzt einige Vorverarbeitungsschritte voraus.

Viele wissenschaftliche Artikel sind nur als PDF erhältlich und müssen zunächst nach ASCII übersetzt werden, wofür viele Tools existieren, jedes mit seinen Vor- und Nachteilen. Grundsätzlich gehen mit der Übersetzung alle Abbildungen in einem Text verloren und auch der Inhalt von Tabellen kann nicht mehr interpretiert werden, da ihre Strukturen nicht ohne Weiteres rekonstruierbar sind. Nach der Übersetzung wird der Text tokenisiert, indem nach Leerzeichen, Tabulatoren und Zeilenenden getrennt wird. Alle Satzzeichen werden zuvor mit Leerzeichen von den Wörtern getrennt\(^1\). Die gesamte Wortfolge eines Textes wird in kleinere Textabschnitte, zumeist Sätze, aufge-

\(^1\)Je nach Anwendung werden auch andere Tokenisierungen vorgenommen.
teilt. Dieser Schritt ist nicht trivial, da nicht an jedem Punkt ein Satz beendet wird. Für ein genaues Erkennen der Satzgrenzen sind daher kontextbezogene oder regelbasierte Verfahren einzusetzen.


3.1 Modelle


Alle Entitäten, die für eine Relation in Frage kommen, werden nach ihrer Reihenfolge im Text indiziert. Ein Klassifikator für eine Relation $R$ in einer Menge von Texten $T$ mit Indexpositionen aus $\mathbb{N}$ ist eine Funktion:

$$C_R : T \times \mathbb{N}^m \mapsto \{1, 0\}. \quad (3.1)$$

3.1.1 Sequenzalignment


Für die Berechnung eines Alignments zwischen zwei Sequenzen wird eine Tabelle mit dem Spaltenvektor der einen und dem Reihenvector der anderen Symbolsequenz vollständig berechnet (Tab. 3.2). An der Tabelle können nun Alignments für alle Teilfolgen aus diesen Sequenzen abgelesen werden. Der Wert einer Tabellenzelle ergibt beispielsweise für ein optimales lokales Alignment zweier Teilsequenzen aus den Sequenzen $t_1$ und $t_2$:

\[
F(i, j) = \max \begin{cases}
0 \\
F(i, j - 1) + s(t_1(i), \text{Gap}) \\
F(i - 1, j) + s(\text{Gap}, t_2(j)) \\
F(i - 1, j - 1) + s(t_1(i), t_2(j))
\end{cases}
\]  \hspace{1cm} (3.2)

Das optimale Alignment wird ausgehend von der Tabellenzelle mit dem höchsten Wert bestimmt, indem ein Pfad über die Zellen zurückverfolgt wird, die zur Berechnung des jeweiligen Vorgängers ($i - 1$ bzw. $j - 1$) herangezogen wurden (Tab. 3.2). Für die Berechnung von $s(a, b)$ wird eine Substitutionsmatrix verwendet, die für zwei Symbole aus dem Sequenzalphabet einen Kostenwert für deren Ersetzung festlegt. Das einfachste Ersetzungsschema bewertet einen Match mit 1 und einen Mismatch mit 0, was einem Ähnlichkeitsmaß analog zum Editabstand entspricht. Im Besonderen sind problemspezifische Substitutionsmatrizen anzuwenden (siehe Kapitel 5.2). Auch für die Funktion $F(i, j)$ finden sich viele Varianten.

Mit der Funktion $\text{consensusTags}$ werden aus einem Text alle Wörter ausgewählt und zu einem neuen Text zusammengefasst, die das gleiche Tag wie
### 3.1 Modelle

#### Tabelle 3.2: Alignment zwischen zwei POS-Sequenzen. Das lokale Alignment nach Zurückverfolgen des eingezeichneten Pfades ergibt:

<table>
<thead>
<tr>
<th></th>
<th>DT</th>
<th>PTN</th>
<th>INT</th>
<th>IN</th>
<th>JJ</th>
<th>PTN</th>
<th>CC</th>
<th>PTN</th>
<th>NN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTN</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>INT</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>IN</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PTN</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>11</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Das ihnen gegenüberliegende Wort in einem Pattern-Text $s$ haben:

$$t_c = consensusTags(s, t)$$  \hspace{1cm} (3.3)

$$= w_j \ldots w_k, \text{ mit } i = j \ldots k \wedge w_i = t(i) \wedge t(i).tag = s(i).tag. \hspace{1cm} (3.4)$$

Nachdem ein Pattern-Text mit einem unbekannten Text aligniert und alle Wörter mit matchenden Tags zu einem neuen Text zusammengefügt sind, werden die darin enthaltenen, nach ihrer Reihenfolge indizierten Entitäten, die in einer Relation stehen, mit einem Klassifikator identifiziert. Der Klassifikator besteht aus einer Menge von Indextupeln, die aus Trainingsbeispielen bekannt sind. Für alle Tupel in dieser Menge werden im Text die entsprechenden Entitäten der Relation $R$ hinzugefügt (Abb. 3.1). Ein Pattern besteht somit aus einem Text $s$ als Sequenz zur Alignierung mit neuen Texten und einem Klassifikator in Form einer Menge $K$ von Indextupeln:

$$pattern = (s, K). \hspace{1cm} (3.5)$$

Wird ein solches Pattern zur Bestimmung einer Relation $R$ in einem Text $t$ verwendet, ist die Klassifikationsfunktion wie folgt definiert:

$$C_R(t_c, i, j, \ldots, k) = 1 \text{ gdw. } t_c = consensusTags(align(s, t)) \wedge (i, j, \ldots, k) \in K. \hspace{1cm} (3.6)$$
Eine der am häufigsten angewendeten Methoden zur Informationsextraktion aus einem Text sind reguläre Ausdrücke, die zumeist manuell erstellt werden und mit denen ein Text nach passenden Abschnitten geparsst wird. In den gefundenen Textabschnitten werden die Entitäten anschließend nach einem vorgegebenen Muster einer Relation zugeordnet. In einer vorangegangenen Studienarbeit wurde eine Menge solcher Ausdrücke parametrisiert und anschließend mittels eines Genetischen Algorithmus eine verbesserte Lösung gefunden [Plake, 2004].

Ein regulärer Ausdruck ist allgemein durch einen endlichen Zustandsautomaten (Finite State Automaton, FSA) repräsentiert, der ein effizientes Text-Parsen erlaubt und einfach parametrisiert werden kann, was die Anwendung eines Genetischen Algorithmus begünstigt, um die Struktur eines solchen Automaten zu erlernen. Abbildung 3.2 zeigt einen Zustandsautomaten für eine gebräuchliche Pattern-Sequenz zur Extraktion von Protein-Protein-Interaktionen. Für einen gegebenen Automaten wird der Text beginnend im Startzustand solange verfolgt, wie eine Transition in einen Folgezustand existiert. Der geparsste Text gehört zur Sprache des Automaten, wenn die Eingabe in einem Endzustand beendet wird. In einem solchen Text werden
3.1 Modelle

Abbildung 3.2: Ein Mealy-Automat zur Erkennung von annotierten Sequenzen in einem Text, die eine Protein-Protein-Interaktion beschreiben.

die relevanten Entitäten indiziert und die Extraktion anhand einer Menge von bekannten oder erlernten Indextupeln durchgeführt. Das Parsen wird der Reihe nach, mit jedem Wort im Text beginnend, wiederholt. Ein Pattern besteht somit aus einem endlichen Zustandsautomaten $M$ und einem Klassifikator in Form einer Menge $K$ von Indextupeln:

$$\text{pattern} = (M, K) \quad (3.7)$$

$$C_R(t, i, j, ..., k) = 1 \text{ gdw. } t \in L(M) \land (i, j, ..., k) \in K. \quad (3.8)$$

3.1.3 Hidden Markov Modelle

Abbildung 3.3: Beispiel für ein Pattern als eine Sequenz von Tags aligniert mit einem Text (A) und als ein Mealy-Automat (B). Der probabilistische Moore-Automat (C) ist ein HMM zur Generierung von Sequenzen, die häufig Protein-Protein-Interaktionen beschreiben. Anhand einer vollständig annotierten Trainingsmenge können eigene Zustände (+) für in Relation stehende Entitäten bestimmt werden.


Ein Pattern besteht in diesem Fall aus einem HMM $\Theta$ und einem Klassifikator in Form einer Menge $K$ von Indextupeln:

$$\text{pattern} = (\Theta, K).$$  \hfill (3.9)

Die Klassifikationsfunktion kann daraufhin wie folgt definiert werden:

$$C_R(t, i, j, ..., k) = 1 \text{ gdw. } P(t|\Theta) > 0 \land (i, j, ..., k) \in K.$$  \hfill (3.10)

In Abbildung 3.3 sind noch einmal alle bisher beschriebenen Pattern-Modelle beispielhaft dargestellt.

### 3.1.4 Vector Space Modelle


$$C_R(t, i, j, ..., k) = 1 \text{ gdw. } C_{\text{SVM}}(\text{featureVector}(t, i, j, ..., k)) > 0.$$  \hfill (3.11)

Als Features dienen z.B. die Wörter und POS-Annotationen, die in der Phrase zwischen den Entitäten und um sie herum enthalten sind, aber auch Chunk- und SPO-Informationen können hierfür herangezogen werden. Ein vektorbasiertes Verfahren wurde mit solchen Features bereits erfolgreich zur Bestimmung von Protein-Protein-Interaktionen angewendet [Xiao et al., 2005].

### 3.2 Lernverfahren

Für die verschiedenen Modelle sollen die Modellparameter aus annotierten Trainingsbeispielen erlernt werden, wofür sich unterschiedliche Verfahren eignen, von denen im folgenden einige genauer beschrieben sind.
Abbildung 3.4: Schema zum Erlernen eines Modells für die Extraktion von Relationen aus einem Text (a). Bei Anwendung eines Pattern-Generating Algorithmus (Kapitel 3.2.1) wird der Zyklus (1) nur einmal durchlaufen, da die Patterns nach einmaligem Aufruf des PGA sofort zur Verfügung stehen. Für die Evaluation werden die vom gelernten Modell aus einem Testkorpus extrahierten Relationen mit der Musterlösung verglichen (b).

Neben den Modellparametern wie Alignmentsequenzen oder Transitions-tabellen für FSAs, sind die Indextupel der in Relation stehenden Entitäten als Menge $K$ zu bestimmen und den Patterns hinzuzufügen. Sollen mehrere Patterns der Reihe nach gelernt werden, wird eine Separate-And-Conquer Strategie angewendet, die alle Trainingsbeispiele entfernt, die von einem erlernten Pattern überdeckt werden (separate) und anschließend auf den übrigen Beispielen ein neues Pattern versucht (conquer), solange kein weiteres Pattern benötigt wird oder alle Beispiele der Trainingsmenge überdeckt sind. Abbildung 3.4 zeigt den Ablauf des Lernprozesses für ein Modell, wie er z.B. bei der Evolution von Zustandsautomaten stattfindet.

### 3.2.1 Patterngenerierung

Die Bestimmung von Beispielsequenzen, die eine Relation im Text beschreiben, kann mit dem Algorithmus in Abbildung 3.5 erfolgen. Als Sequenzen für die Eingabe dienen vollständig annotierte Textabschnitte, die alle relevanten
Input: Menge von annotierten Sequenzen $S$, Schwellwert $d$
Output: Menge $P$ von Patterns

1. Für alle $s_i, s_j \in S$ mit $(i \neq j)$:
   a) $(s_{ia}, s_{jb}) = align(s_i, s_j)$
   b) $c_{sj} = consensusTags(s_{ia}, s_{jb})$
      $c_{si} = consensusTags(s_{jb}, s_{ia})$
   c) Füge Indizes der in Relation stehenden Entitäten in $c_{si}$ und $c_{sj}$
      einer Menge $K$ hinzu und wähle Pattern $p = (c_{si}, K)$ oder $p =$
      $(c_{sj}, K)$.
   e) Wenn $p \in P$, erhöhe den Zähler für $p$ um eins, ansonsten füge $p$
      der Menge $P$ mit Zählerstand eins hinzu

2. Für alle $p \in P$: Entferne $p$, wenn der Zählerstand kleiner ist als $d$

3. Gebe $P$ aus

Abbildung 3.5: Der Pattern-Generating Algorithmus (PGA) adaptiert aus
[Huang et al., 2004].

Entitäten und Relationsbeschreibungen enthalten. Der Pattern-Generating
Algorithmus (PGA) sollte ein lokales Alignment durchführen, wenn die rele-
vanten Abschnitte innerhalb der Trainingssequenzen eher kurz sind.

Die Werte der Substitutionsmatrix sind statistisch aus ideellen Alignments
abzuleiten. Liegt kein alignierter Korpus vor, was meistens der Fall sein wird,
werden am Editabstand orientierte Werte verwendet, mit einer Präferenz
für ein Alignment von Entitätswörtern, die zu einer gemeinsamen Klasse
gehören.

Die Relationsbeschreibungen werden aus allen Trainingssätzen mit einem
gewissen Rand an Wörtern herauskopiert und dienen als Eingabesequen-
zen, aus denen der Algorithmus, zusammen mit einem Schwellwert für die
minimale Auftrittshäufigkeit, neue Patterns erzeugt. Die generierten Pattern-
Sequenzen können direkt mit neuen Texten aligniert oder in Mealy-Automaten
überführt werden, um damit einen Text zu parsen und zusätzlich mit Hilfe
eines Genetischen Algorithmus die Automaten weiter zu optimieren (Kapitel
3.2.3).
3.2.2 Stochastische Optimierung

Das Ziel einer stochastischen Optimierung ist eine Modellstruktur zu finden, mit der die Wahrscheinlichkeit des Modells $M$ für eine gegebene Trainingsmenge $D$ maximiert. Dieser Zusammenhang ist durch den Satz von Bayes ausgedrückt:

$$P(M|D) \propto P(D|M) \times P(M).$$


Im Allgemeinen lassen sich Expectation Maximization (EM) Verfahren anwenden, die den Erwartungswert eines probabilistischen Modells für eine Menge von Trainingsdaten nach einem iterativen Schema durch Anpassung der Modellparameter maximieren. Siehe dazu auch die Arbeit von Xiao et al. zur Extraktion von PPI aus Texten mit einem probabilistischen, vektorbasierten Verfahren [Xiao et al., 2005].

3.2.3 Genetische Algorithmen


Allgemein läßt sich ein GA in folgenden Pseudo-Code übersetzen:

1. Wähle eine geeignete Kodierung der Individuen.
2. Initialisiere eine zufällige Population von Individuen und nenne die Ausgangspopulation Generation 0.


5. Mutiere alle Nachkommen gemäß Mutationsschema.


7. Wenn Abbruchbedingung nicht erfüllt ist, gehe zu 3.

8. Gebe das beste Individuum aus.

Kapitel 4

Software


4.1 Tools


Eine Textmining-Pipeline


Konform mit der Definition 2.1 wurde eine Klasse `Word` erstellt, die einen String Token und einen String Tag umfasst. Eine Folge von Wörtern ist mit der Klasse `Phrase` repräsentiert (Def. 2.2). Schließlich ist mit der Klasse `Entity` eine textuelle Entität beschrieben, die ebenfalls dem Context hinzugefügt werden kann. Eigenschaften der Klasse umfassen einen Namen aus einer Wortfolge sowie einen ID-String, mit dem die Entität zusätzlich referenziert werden kann. Eine Menge von Word-Objekten kann in der Klasse `Alphabet` zusammengefasst werden, die jedem Wort durch die `indexOf`-Methode seinen Index im Alphabet zuordnet, wenn es darin enthalten ist. Basierend auf diesen Klassen wurden ein `DFA`, ein zugehöriger Parser sowie Algorith-
men für globales, lokales und End-Space-Free Alignment implementiert und
die Funktionalität der einzelnen Methoden anhand von JUnit-Testfällen si-
chergestellt.

Eine weitere wichtige Klasse ist der *Splitter*, der eine Wortfolge an Wörtern
 teilt, die über einen regulären Ausdruck identifiziert werden und dadurch
 einen Text in einzelne Passagen oder Sätze partitioniert. Für jede Partiti-
on wird ein eigener *Context* erzeugt, der dann mit verschiedenen Pipeline-
 Komponenten in einer festgelegten Reihenfolge verarbeitet wird. Wie be-
 reits in der Studienarbeit [Plake, 2004] wird eine Entity-Mapper-Komponen-
te eingesetzt, um Entitäten anhand von Synonymklassen zu normaliern
(Def. 2.5). Eine Entität wird über ihren Namen mit allen bekannten Synony-
men aus allen Klassen paarweise (global) aligniert und bekommt die ID der
ähnlichsten Synonymklasse zugewiesen. Im Kapitel 5.4 werden Komponen-
ten für eine konkrete Anwendung eingeführt, darunter der in 4.1 beschrie-
bene Protein-Tagger, ein Tagger für Interaktoren, die den Typ einer Inter-
 aktion angeben, und mehrere Extraktionsskomponenten für Protein-Protein-
Interaktionen. Der TnT-Tagger und der Shallow-Parser sind nicht als Java-
anwendungen verfügbar, weshalb auch keine entsprechenden Pipeline-Kom-
 ponenten existieren. Die POS-Annotation und eventuell das Chunking und
Shallow-Parsing wird für jeden Text extern einmalig durchgeführt und jeweils
in einer Datei gespeichert. Diese Informationen können, wie z.B. in Abbildung
4.1, während der Preprocessing-Phase geladen und dem Context hinzugefügt
werden. Die Ausgabe des Verfahrens ist typischerweise eine Textdatei, in
die am Ende der Pipeline alle (gesammelten) Ergebnisse hineingeschrieben
werden.

Um das Programmieren im Rahmen der Pipeline zu vereinfachen, eine
weitere Verwendung zu motivieren und schließlich eine Solidität der geschrie-
benen Software zu erreichen, wurden alle Klassen und Methoden ausführ-
lich dokumentiert. Der Sourcecode und die Java-Dokumentation zur Pipeli-
 ne sind, zusammen mit JUnit-Testfällen und einer Beispielanwendung (siehe
Kapitel 5), auf der beiliegenden CD enthalten. Die JUnit-Testfälle stammen
u.a. aus verschiedenen Büchern und Skripten zum Thema endliche Automates
und Sequenzalignment [Durbin et al., 2002, Gusfield, 1999].
Kapitel 5

Anwendung: Protein-Protein-Interaktionen


In diesem Kapitel werden die eingeführten Methoden an der praxisnahen Anwendung der Erkennung von PPI evaluiert und miteinander verglichen. Die praktische Umsetzung basiert auf den im Kapitel 4 vorgestellten Softwarekomponenten, sowie weiteren anwendungsspezifischen Implementatioen, die in Abschnitt 5.4 ausführlich beschrieben sind. Die Evaluation wird im wesentlichen anhand einer Menge von Trainingssätzen, zum einen aus dem BioCreative Corpus (BCC) und aus dem IEPA-Korpus [Berleant et al., 2003], durchgeführt. Die Sätze im BCC wurden per Hand auf PPI geprüft und die


Im Folgenden werden die anwendungsspezifischen Entitäten und Relationen definiert.

Sei $P$ die Menge aller Wörter, die Bestandteil eines Proteinnamens sind:

$$ P = \{w = (\text{token}, \text{tag})| \text{tag} = \text{PTN}\} \quad \text{(5.1)} $$
und \( p \) eine Entität, die ein Protein bezeichnet:

\[
p = (w_0, w_1, \ldots, w_k) \text{ mit } w_i \in P \text{ für } i = 0 \ldots k.
\] (5.2)

Eine Unterscheidung zwischen Gen- und Proteinbezeichnern wird nicht getroffen, da diese oftmals synonym verwendet werden. Weil alle Tags für einen Proteinnamen identisch sind, ersetzen wir zusammengesetzte Namen durch ein einzelnes Wort:

\[
p = ((\text{token}_0, \text{PTN}), \ldots, (\text{token}_k, \text{PTN}))
\] (5.3)

\[
= (\text{token}_0 + \sqcup \ldots \sqcup + \text{token}_k, \text{PTN}).
\] (5.3)

Die Tokens der einzelnen Wörter werden über ein Leerzeichen zu einem einzigen Token konkateneriert.

Die Menge aller Interaktionswörter sei:

\[
I = \{ w = (\text{token}, \text{tag}) | \text{tag} = *\text{INT}* \}.
\] (5.4)

Die * Symbole bedeuten, dass der tag weitere Annotationen enthalten kann, z.B. ob es sich um ein Verb oder ein Substantiv handelt.

Ein Interaktor \( i \) ist eine Entität, bestehend aus einem einzelnen Wort:

\[
i \in I.
\] (5.5)

Eine Interaktion ist ein Tripel aus zwei Proteinen und einem Interaktor:

\[
\text{int} = (p_1, p_2, i).
\] (5.6)

5.1 Vorverarbeitung


Mit dem Tags’n’Treigrams (TnT) [Brants, 2000], einem HMM-basierten POS-Tagger trainiert auf die englische Sprache, wird jedem Token in einem Satz sein Tag zugewiesen. Die POS-Tags sind konform mit dem PennTreebank-Tagset [Marcus et al., 1993]. Der Tag-String wird beim Schreiben in eine Datei dem Token mit einem trennenden /-Zeichen angefügt. Die Proteinbezeichner werden mit einem speziellen Tool [Bickel et al., 2004] annotiert und dabei die POS-Tags der zugehörigen Tokens durch das Tag ’PTN’ ersetzt. In einem letzten NER-Schritt werden die Interaktoren anhand einer Wortstammliste bekannter Begriffe identifiziert (Tab. 5.2). Alle Wörter, ausgenommen den Proteinnamen, werden auf ihre Stammform zurückgeführt und mit den Wortstängen in der Liste verglichen. Bei einem Token mit positiver Identifikation wird dem zugehörigen Tag ein ’INT’ angefügt. Die Wortstammbildung wurde mit einem einfachen regelbasierten Stemmer (Porter-
5.1 Vorverarbeitung

<table>
<thead>
<tr>
<th>abrog</th>
<th>acetyl</th>
<th>activ</th>
<th>affect</th>
<th>apparatu</th>
<th>associ</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind</td>
<td>block</td>
<td>bound</td>
<td>cluster</td>
<td>complex</td>
<td>compon</td>
</tr>
<tr>
<td>compris</td>
<td>conjug</td>
<td>contact</td>
<td>contain</td>
<td>control</td>
<td>convers</td>
</tr>
<tr>
<td>deriv</td>
<td>destabil</td>
<td>downregul</td>
<td>effect</td>
<td>encod</td>
<td>enhanc</td>
</tr>
<tr>
<td>exhibit</td>
<td>express</td>
<td>form</td>
<td>format</td>
<td>fuse</td>
<td>hyperexpress</td>
</tr>
<tr>
<td>includ</td>
<td>increas</td>
<td>induc</td>
<td>induct</td>
<td>inhibit</td>
<td>inhibitor</td>
</tr>
<tr>
<td>interact</td>
<td>interf</td>
<td>ligand</td>
<td>link</td>
<td>modul</td>
<td>obstruct</td>
</tr>
<tr>
<td>phosphoryl</td>
<td>potenti</td>
<td>recogn</td>
<td>recruit</td>
<td>regul</td>
<td>repress</td>
</tr>
<tr>
<td>repressor</td>
<td>requir</td>
<td>respons</td>
<td>sever</td>
<td>stabil</td>
<td>stimul</td>
</tr>
<tr>
<td>suppress</td>
<td>suppressor</td>
<td>synthesi</td>
<td>upregul</td>
<td>yield</td>
<td></td>
</tr>
</tbody>
</table>

Tabelle 5.2: Liste der Wortstämme zu allen bekannten Interaktoren.


Jeder Text wird letztendlich komprimiert, d.h. alle aufeinanderfolgenden Wörter mit identischen Tags werden zu einem Wort zusammengefügt. Aus mehreren Zahlwörtern mit dem POS-Tag 'CD' oder einer Adjektiv-Phrase mit mehreren 'JJ'-Tags wird so ein einziges Wort, dessen Token aus den mit Leerzeichen konkatenierten Tokens der einzelnen Wörter zusammengesetzt ist.

Die bis hier besprochenen Vorverarbeitungsschritte bilden die Grundform der Texte, aus denen im Weiteren die PPI extrahiert werden. Eine ergänzende Möglichkeit der Vorverarbeitung, auf die in dieser Arbeit jedoch nicht genauer eingegangen wird, besteht in der Durchführung eines Chunking, bei dem die Wörter eines Satzes zu höheren grammatikalischen Einheiten, wie Nominal- und Verbalphasen, zusammengefasst werden. Ein Text wird weiter vereinfacht, fügt man auch die Wörter einer Chunk-Phrase zu einem einzigen Wort zusammen. Werden in den Sätzen auch Subjekt-Objekt Beziehungen bestimmt, kann diese Information einem Substantiv oder Nomen durch Anfügen einer 'SUBJ'- bzw. 'OBJ'-Zeichenkette an das entsprechende Tag hinzugefügt werden. Im verwendeten Alphabet sind dann alle möglichen Tags, ebenso wie die der Chunks, zu berücksichtigen.

Im Folgenden werden bei der Extraktion von PPI nur Sätze untersucht,
in denen mindestens zwei Proteine und ein Interaktor markiert sind. Die meisten Tokens der Wörter in dem Alphabet in Tabelle 5.1 sind leere Strings, die nicht mit anderen Tokens verglichen werden, so daß in diesen Fällen die Entscheidung beim Alignment oder Parsen allein von der Annotation in Form der Tag-Zeichenkette abhängig gemacht wird.

### 5.2 Pattern-Alignment

In dem ersten untersuchten Verfahren werden Patterns für die PPI-Extraktion mit Hilfe des Pattern-Generating Algorithmus (PGA) aus Kapitel 3.2.1 generiert. Das Alphabet der Symbolsequenzen besteht aus den POS- und NER-Tags der Wörter, die während der Vorverarbeitung des Textes annotiert wurden (Tab. 5.1). Die Benutzung eines Alignment-Algorithmus erfordert eine Substitutionsmatrix oder Kostenfunktion, die je zwei Symbole aus dem Sequenzalphabet einen Match- bzw. Mismatch-Score zuweist. Tabelle 5.3 zeigt die in diesem Ansatz verwendete Substitutionsmatrix für alle relevanten POS- und NER-Tags. Die Kosten für die Ersetzung von Wörtern mit Tags für Proteine und POS mit einem Gap sind aus einer anderen Arbeit zu diesem Thema übernommen [Huang et al., 2004]. Die Substitutionswerte für Wörter mit zwei verschiedenen Tags sind mit Ausnahme der Entitätsbezeichner für Proteine und Interaktoren gleich 0. Für zwei Symbole aus einer der Entitätsklassen Protein oder Interaktor ist der Match-Score etwas höher.
Tabelle 5.4: Generierte Pattern-Sequenzen aus dem BioCreative Corpus.

<table>
<thead>
<tr>
<th>Support</th>
<th>Sequenz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1405</td>
<td>PTN VBINT PTN</td>
</tr>
<tr>
<td>258</td>
<td>PTN VBINT DT PTN</td>
</tr>
<tr>
<td>173</td>
<td>PTN VBINT IN PTN</td>
</tr>
<tr>
<td>138</td>
<td>PTN NNINT PTN</td>
</tr>
<tr>
<td>116</td>
<td>PTN NNINT IN PTN</td>
</tr>
<tr>
<td>46</td>
<td>PTN VBINT IN DT PTN</td>
</tr>
<tr>
<td>45</td>
<td>PTN RB VBINT PTN</td>
</tr>
<tr>
<td>35</td>
<td>NNINT IN PTN IN PTN</td>
</tr>
<tr>
<td>35</td>
<td>PTN VBINT NN PTN</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>PTN VBINT TO PTN</td>
</tr>
<tr>
<td>7</td>
<td>PTN CC PTN VBINT PTN</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>5</td>
<td>PTN ( PTN ) VBINT PTN</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>PTN VB DT NNINT IN PTN</td>
</tr>
<tr>
<td>1</td>
<td>PTN VBINT NN PTN CC PTN</td>
</tr>
<tr>
<td>1</td>
<td>PTN ( PTN ) NNINT NN VBINT PTN</td>
</tr>
</tbody>
</table>

damit sich gleiche Entitäts-Tags bei der Alignierung eines Patterns mit einem Text nach Möglichkeit gegenüberstehen, da nur diese Wörter aus dem Text von der Consensus-Funktion (Gleichung 3.3) in einen neuen Text übernommen und in ihm schließlich alle zum Pattern bekannten Relationen bestimmt werden.

Die Trainingssätze oder Phrasen, die eine PPI beschreiben, bilden eine Menge von Beispielsequenzen, aus denen der PGA die Patterns generiert. Enthält die Musterlösung zu den Namen der interagierenden Proteine und Interaktoren die passenden Token-Indizes aus dem Trainingssatz, kann jede Interaktionsbeschreibung aus diesem herauskopiert werden, indem alle Wörter vom Protein/Interaktor mit dem kleinsten Token-Index bis zum Protein/Interaktor mit dem größten Token-Index zu einem neuen Text zusammengefasst werden. Zusätzlich kann um die gesamte Phrase ein Rand aus benachbarten Wörtern belassen werden.

In den 1000 Sätzen des BioCreative Corpus (BCC) finden sich 256 in-
teraktionsbeschreibende Phrasen, aus denen der PGA bei einem Schwellwert von 1.0 für die minimale Auftrittshäufigkeit 155 Patterns generiert. Tabelle 5.4 zeigt einige der generierten Patterns, sortiert nach der Häufigkeit ihrer Entstehung. Alle Wörter wurden nur POS- und NER-annotiert, keine weiteren Annotationen von Chunkern oder Shallow-Parsern sind hier verwendet. Die generierten Patterns können, wie in 3.1.1 beschrieben, mit neuen Sätzen aligniert und die Relationen anhand der Indextupel extrahiert werden. An den drei verschiedenen Korpora BCC, IEPA und DIP wurden Qualitätsmerkmale wie Precision und Recall gemessen und die Ergebnisse in 5.5 und 5.6 für diesen Ansatz präsentiert und genauer untersucht.

5.3 Endliche Zustandsautomaten

In einem zweiten Verfahren werden Patterns als Zustandsautomaten erlernt, die reguläre Ausdrücke repräsentieren und ein effizientes Parsen des Textes erlauben (Kapitel 3.1.2).

Vom Pattern-Generating Algorithmus (PGA) generierte Pattern-Sequenzen lassen sich in einen Mealy-Automaten überführen, indem, beginnend beim ersten Wort, solange in einen neuen Zustand geschaltet wird bis die Sequenz zu Ende ist. Der letzte Zustand wird als Endzustand markiert. Die Menge $K$ an Indextupeln, die der PGA zu jedem Pattern bestimmt hat, kann für die Automaten ohne Änderung übernommen werden. Um diese Patterns weiter zu generalisieren, können variable Wortabstände zwischen den einzelnen Entitätstransitionen eingefügt werden (Abb. 5.2). Um die Struktur eines Mealy-Automaten komplett zu erlernen, werden alle Parameter des Automaten mit einem Genetischen Algorithmus (GA) für eines der Qualitätsmerkmale Precision oder Recall optimiert. Ein Automat ist gegeben als ein Tupel $M = (Z, \Sigma, \delta, z_0, E)$. Die Anzahl der Zustände in $Z$ wird vorab festgelegt. Zu erlernen sind die Transitionsfunktion $\delta$ und die Menge der Endzustände $E$. Die Transitionsfunktion ist wie in Abbildung 3.2 als eine Zustandsübergangstabelle für die Symbole aus dem Alphabet $\Sigma$ repräsentiert.

Der Wert jeder Tabellenzelle wird der Reihe nach in einen binären Vektor hinterlegt und diesem am Ende ein Bit pro Zustand hinzugefügt, um die
5.3 Endliche Zustandsautomaten

Abbildung 5.2: Ausschnitt aus einer generierten Patternsequenz mit einer Transition für ein Protein-Tag (A). In den unteren Bildern wurde die Transitionsstruktur durch das Zulassen von Zwischenwörtern weiter generalisiert. Der Automat in (B) akzeptiert nacheinander unendlich viele POS-Tags, bevor ein Protein kommen muss, während der Automat in (C) höchstens zwei solcher Zwischenwörter zulässt.

Menge der Endzustände zu kodieren. Für einen Automaten mit 7 Zuständen ergeben sich 3 Bit pro Zelle um 8 mögliche Transitionen, inklusive einer Null-Transition, zu kodieren. Bei einem Alphabet mit 19 Symbolen wird der Automat mit $7 \times 19 \times 3 = 399$ Bit für die Transitionstabelle und 7 Bit für die Endzustände vollständig beschrieben.

Nach Transformation der vom PGA generierten Pattern-Sequenzen in Mealy-Automaten, können diese durch einen GA weiter optimiert werden. Geht es nur um das Lernen von variablen Wordgaps zwischen den Entitästransitionen, um die Generalität des Automaten für das Parsen mit neuen Texten zu erhöhen, sollten nur die Abstandsparameter kodiert und beim Einsetzen der Werte die Struktur des Automaten automatisch angepasst werden, wie in Abbildung 5.2 angedeutet.

Wird ein Pattern mit einem Automaten von Grund auf mit dem GA neu erlernt, müssen auch die Indextupel der in Relation stehenden Entitäten kodiert und erlernt werden, wie es analog mit der Transitionstabelle geschieht. Für jede Entität in der Relation besitzt die Tabelle eine Dimension aus $N$ mit einer fest begrenzten Anzahl an möglichen Indizes. Jeder Tabelleneintrag ist mit einem Bit für Ja oder Nein kodiert und bedeutet, ob die Entitäten hinter den entsprechenden Indizes miteinander in Relation stehen oder nicht. Bei der Extraktion ergeben sich bei maximal 4 Proteinen und 2 Interaktoren $2 \times (4 \times (4 - 1))/2 = 12$ mögliche Interaktionen, wenn die Richtung zwi-
schen den Proteinen symmetrisch und keine Selbstinteraktionen zugelassen sind. Soll auch die Interaktionsrichtung zwischen den Proteinen erlernt werden, wird jedem Paar ein Richtungswert angefügt, anstatt zwei verschiedene Entitätstupel zu kodieren.

Die Evolution der Patterns mit einem Genetischen Algorithmus erfolgt mit Hilfe des in Kapitel 4 vorgestellten Programms, das bereits ein einfaches Evolutionsschema für binäre Vektoren anbietet. Nach dem Schema werden anhand der Fitness die besten Individuen einer Generation (Elite) selektiert, die dann neue Nachkommen erzeugen, die sich durch One-Point-Crossover und Mutation von ihren Eltern unterscheiden.

Das Evolutionsschema und seine Parameter wurden während der durchgeführten Experimente und der Evaluation nicht verändert. Die Selektionsrate beträgt immer ein Viertel (25%) der Population, in der sich konstant 20 bzw. 40 Individuen befinden\(^1\). Die Mutationsrate ist auf 10% festgelegt, d.h., dass im Schnitt jedes 10. Bit der Sequenz eines Individuums nach seiner Erzeugung umgekippt wird.


Bei Verwendung einer Precision als Fitnessmaß wird der Wert mit der Anzahl der gefunden true Positives (TPs) gewichtet. Das Optimum von 1 wird in diesem Fall nie erreicht, sondern steigt, je mehr TPs gefunden wurden, um den trivialen Fall von einem TP pro Regel zu vermeiden:

\[
fitness = (1 - 1/TP) \times Precision, \text{ falls } TP > 0. \tag{5.7}
\]

In Abbildung 5.3 sind die beiden angewendeten Verfahren als Extraktionskomponenten visualisiert.

\(^1\)Mit beiden Populationsgrößen wurden identische Ergebnisse erzielt.
5.4 Implementation

Die Anwendung zur Extraktion von Protein-Protein-Interaktionen aus Texten wird innerhalb des in Kapitel 4.2 eingeführten Frameworks als konkrete Textmining-Pipeline, mit gegeneinander austauschbaren Komponenten für jede zu testende Methode, realisiert.

**Package wbi.ie.tmpipe**

te zur Repräsentation von Texten durch Features und Feature Vectors und schließlich VSM Classifier wie Nearest-Neighbor und Naive-Bayes sowie einen Wrapper für die Support Vector Machine LIBSVM [Chang and Lin, 2001].

**Package wbi.ie.tmpipe.tasks.ppi**


Für die Klassifikation und Extraktion von Interaktionen wurde eine Klasse *Interpreter* als eine Tupelmenge von Protein- und Interaktor-Indizes und eine Methode *interpret* implementiert, die aus einer Phrase, in der mindestens die Proteine und Interaktoren mit 'PTN' bzw. 'INT' annotiert sind, alle PPI entsprechend der Menge der Indextupels zurückgibt. Um am Interpreter eine PPI anzumelden wird die Methode *addInteraction* aufgerufen und ihr die Indizes der beide Proteine, des Interaktors und ein Wert
für die Interaktionsrichtung übergeben. Die Basisklasse *Pattern* enthält eine Referenz auf einen *Interpreter* und bietet für diesen eine *set*- und *get*-Methode an. Als spezielle Implementation für ein Pattern befindet sich ein *DfaPattern* im Package *wbi.ie.tmpipe.tasks.ppi.fsa* und ein *AligningPattern* in *wbi.ie.tmpipe.tasks.ppi.align*.

In diesem Package befindet sich auch die Extraktionskomponente *HandcraftedExtractor* aus einer vorangegangenen Studienarbeit, die mit einer XML-Regeldatei initialisiert wird, in der die Patterns nach einem vorgegebenen Schema definiert sind.

**Package wbi.ie.tmpipe.tasks.ppi.ner**

Die Identifikation von Proteinbezeichnern in einem natürlichsprachlichen Text ist für sich eine schwierige Aufgabe. Für die Protein-Annotation ist ein SVM-basierter Klassifikator mit einem regelbasierten Postprocessing Schritt für die Erkennung zusammengehöriger Wörter eines Proteinnamens als Pipeline-Komponente verfügbar. Der *ProteinTagger* wurde auf einer Menge von 15000 Sätzen trainiert und kann nun beliebige Wortfolgen verarbeiten, wobei auf eine Tokenisierung konform mit den Trainingsdaten des Taggers zu achten ist. Alle aufeinander folgenden Proteinwörter ergeben ein Protein, das im *PPI-Context* entsprechend markiert wird. Die Proteinwörter bekommen zudem das Tag 'PTN' zugewiesen.


**Package wbi.ie.tmpipe.tasks.ppi.eval**

Für eine automatisierte Evaluation stellt dieses Package weitere Klassen zur Verfügung. Ein Korpus besteht aus einer Menge von tokenisierten Sätzen in


Mit der statischen Methode *load* der Klasse *Evaluator* kann ein neues Objekt dieser Klasse zu einem gegebenen Goldstandard erstellt werden. Die wichtigste Methode des *Evaluator* ist *evaluate*, die eine Hashtable entgegennimmt, in der jeweils eine Satz-ID auf eine Liste von Interaktionen verweist, die in diesem Satz gefunden wurden. Durch einen Vergleich der Interaktionen mit den Musterlösungen werden in jedem Satz die *true Positives* (TPs) und *false Positives* (FPs) bestimmt. Doppelte TPs werden nur einmal gezählt. Die TPs und FPs werden aus allen Sätzen aufsummiert. Die *false Negatives* (FNs) ergeben sich aus der Menge aller Interaktionen, ausgenommen den TPs. Nach Beendigung der Methode werden die Werte für TP, FP, FN, *Precision* und *Recall* an den Aufrufer zurückgegeben.

5.4 Implementation

beispiele entfernt. Anschließend wird ein neuer zufälliger Pool generiert und auf den verbleibenden Beispielen evolviert (Separate-and-Conquer).

Der DfaPool ist ein GenePool und damit ebenso ein eve.ga.GeneEvaluator, der eine Fitnessfunktion anbietet, um Gene, die in diesem Fall Parameter eines DfaPattern kodieren, nach ihrem Extraktionsergebnis auf einem Testkorpus zu bewerten. Der GeneEvaluator wird vom Populations-Manager aufgerufen, der mit dessen Methoden evaluate und getFitness die Individuen in seinem Pool der Reihe nach bewertet.


Analog gibt es auch ein startfähiges Programm, um eine DIFevaluation durchzuführen. Dem Programm wird neben der zu wählenden Komponente ein Verzeichnis übergeben, worin die Texte nach einem vorgegebenen Muster POS- und Protein-annotiert hinterlegt sein müssen. Über die in ppi.properties angegebene Synonymdatei wird eine EntityMapper-Komponente geladen, die nach Aufruf der Methode process alle im Context-Objekt gespeicherten Entitäten auf eindeutige IDs abbildet. Die Komponente kann mit einem Alignment-Algorithmus initialisiert werden, der jedoch bisher ausgeblendet wird. Um die Laufzeit mit einer großen Synonymliste\(^2\) zu erhöhen, wird ein einfacher Similarity-Algorithmus ohne Gaps verwendet. Das Ergebnis wird in einem Vorverarbeitungsschritt einmalig ermittelt, bevor alle Texte

\(^2\)Die verwendete Liste zu den Hefe-Proteinen umfasst 7928 Einträge und 14778 Synonyme.
nach den PPI durchsucht werden. Die Menge der extrahierten Interaktionen wird zum Schluss mit der Liste der gesuchten verglichen, die Anzahl der gefundenen bestimmt und neben weiteren Informationen auf der Konsole ausgegeben.

**Package wbi.ie.tmpipe.tasks.ppi.util**

Häufig wiederkehrende Aktionen, wie Dateien und Texte im BioCreative-Format zu laden, in einen PPIContext zu transformieren oder einzelne Proteine oder Interaktoren in einer Phrase zu selektieren, sind als statische Methoden in der Klasse *BCToolbox* implementiert.


Die Klasse *IEPAToolbox* bietet statische Methoden zur Erstellung von Text- und Golddateien im BCC-Format aus den exportierten Exceltabellen des IEPA-Korpus [Berleant et al., 2003].

**Package wbi.ie.tmpipe.tasks.ppi.align**

Dieses Package enthält alle Klassen, um eine Extraktion mit alignierten Patterns als Pipeline-Komponenten zu realisieren. Die Ersetzungskosten für zwei Symbole des Alphabets wird mit einer Matrix bestimmt, die jedem Wortpaar einen Score zuweist. Die Klasse *PPIMatrix* überschreibt die Methode *getScore* der Klasse *Substitutionsmatrix* wie folgt (nicht gezeigt sind die Gap-
Ein AligningPattern erweitert die Klasse Pattern um eine Phrase, die mit anderen Phrasen aligniert werden kann. Die Klasse PatternGenerator implementiert den PGA aus Kapitel 3.2.1 durch die statische Methode generatePatterns, die dem Aufrufer ein Array von AligningPattern zurückgibt.


**Package wbi.ie.tmpipe.tasks.ppi.fsa**

Dieses Package enthält Klassen zur Extraktion von PPI mit endlichen Zustandsautomaten. Ein DfaPattern besitzt neben dem Interpreter eine Referenz auf einen DFA. Der DfaExtractor parst und interpretiert die Phrase
Anwendung: Protein-Protein-Interaktionen

eines Context-Objektes mit seinen Patterns und fügt die vom Interpreter extrahierten Interaktionen dem PPIContext hinzu, wenn sie darin noch nicht enthalten sind.


Für die Nutzung einer externen HMM-Bibliothek wurde zudem die Klasse HmmHelper implementiert, die eine statische Methode createMaxSpecificHmm zur Erstellung eines HMM-Objektes der Java-Bibliothek Jahmm aus einer Menge von Trainingsphrasen enthält. Für jedes Wort einer Trainingsphrase wird ein eigener Zustand erzeugt, der genau den Tag des Wortes emittiert. Die Zustände der gesamten Phrase bilden einen Pfad zwischen einem Startzustand und einem Endzustand. Die Klasse HmmHelper bietet zudem die Methoden neighborMerge und vMerge an, um die Struktur des HMM zu vereinfachen, ohne das eines der ursprünglichen Trainingsbeispiele nicht mehr generiert werden kann.

Zu den wichtigsten Klassen und Methoden befinden sich in den Packages wbi.ie.tmpipe.junit und wbi.ie.tmpipe.tasks.ppi.junit JUnit-Testfälle, die alle einzeln oder gemeinsam über die Klasse RunAll gestartet werden können und eine grundlegende Funktionalität sicherstellen. Zu allen implementierten Klassen ist eine Java-Dokumentation vorhanden, die auf der beiliegenden CD enthalten ist. Im Anhang dieser Arbeit befinden sich zwei Sequenz-Diagramme zu den main-Methoden der Klassen BCEvaluation und DIPEvaluation.
5.5 Evaluation

Für eine Evaluation der eingesetzten Verfahren stehen drei verschiedene Text-Korpora zur Verfügung. Die präsentierten Ergebnisse sind für jeden Korpus einzeln im Verhältnis 2 zu 1 von Trainings- zu Testmenge zehnfach cross-validiert. Die Richtung zwischen den interagierenden Proteinen wurde nicht verglichen. In der Tabelle 5.5 sind die wichtigsten Merkmale und am Ende in Abbildung 5.10 die Ergebnisse zu jedem Korpus zusammengefasst.

Die Güte des Extraktionsverfahrens wird mit den Größen Precision, Recall und F-Measure gemessen. Zwei Interaktionen gelten als identisch, wenn die Proteinbezeichner, der Typ und die Richtung übereinstimmen\(^3\). Besitzen zwei zu vergleichende Proteine jeweils eine ID, so werden die Proteinnamen ignoriert und die Gleichheit anhand der Übereinstimmung der IDs bestimmt. Sei \(\mathcal{I}\) die Menge aller Interaktionen in einem Text und \(\mathcal{R}\) die Menge an Interaktionen, die aus dem Text extrahiert wurden, dann gilt:

\[
\begin{align*}
\text{Precision} & = \frac{|\mathcal{R} \cap \mathcal{I}|}{|\mathcal{R}|} \\
\text{Recall} & = \frac{|\mathcal{R} \cap \mathcal{I}|}{|\mathcal{I}|} \\
F-\text{Measure} & = \frac{(1 + \beta) \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \text{ mit } \beta = 1
\end{align*}
\]

5.5.1 Der BioCreative Corpus

Der BioCreative Corpus (BCC) umfasst 1000 Sätze aus Abstracts biomedizinischer Artikel - einer zufälligen Auswahl aus insgesamt 15000 Sätzen des BioCreative-Evaluationskorpus -, in denen bereits alle Proteine manuell vorgegeben sind [Blaschke et al., 2004]. In dieser Auswahl wurden per Hand satzweise alle direkt beschriebenen Protein-Protein-Interaktionen (PPI) annotiert. Das interaktionsumschreibende Wort, meistens ein Verb, manchmal ein Substantiv, wurde als Interaktor der PPI zugeordnet. Zu allen Interaktionen wurde in einem Goldstandard ein entsprechender Eintrag vorgenommen

\(^3\)Die Groß-/Kleinschreibung wird beim Vergleich des Typs ignoriert.
Anwendung: Protein-Protein-Interaktionen

<table>
<thead>
<tr>
<th></th>
<th>BCC</th>
<th>IEPA</th>
<th>DIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sätze</td>
<td>1000</td>
<td>489</td>
<td>10777</td>
</tr>
<tr>
<td>Sätze mit PPI</td>
<td>174</td>
<td>304</td>
<td>?</td>
</tr>
<tr>
<td>Proteins</td>
<td>1505</td>
<td>1071</td>
<td>15567</td>
</tr>
<tr>
<td>Interactors</td>
<td>1035</td>
<td>1009</td>
<td>47363</td>
</tr>
<tr>
<td>PPI</td>
<td>256</td>
<td>411</td>
<td>≥297</td>
</tr>
<tr>
<td>ØProteins</td>
<td>1.505</td>
<td>2.203</td>
<td>1.444</td>
</tr>
<tr>
<td>ØInteractors</td>
<td>1.035</td>
<td>2.076</td>
<td>0.439</td>
</tr>
<tr>
<td>ØPPI</td>
<td>0.256</td>
<td>0.845</td>
<td>≥0.002</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P–P–I</td>
<td>10</td>
<td>–</td>
</tr>
<tr>
<td>P–I–P</td>
<td>200</td>
<td>–</td>
</tr>
<tr>
<td>I–P–P</td>
<td>46</td>
<td>–</td>
</tr>
</tbody>
</table>

Tabelle 5.5: Ausgewählte Merkmale der drei Evaluationskorpora. Die Anzahl der Tripelkombinationen konnte im IEPA-Korpus aufgrund fehlender Indizes nicht automatisch gezählt werden.

(Tab. 5.6). In der Tabelle 5.5 sind einige statistische Merkmale für diesen Korpus aufgelistet.

Abbildung 5.4 zeigt die Ergebnisse für ein Alignment mit Pattern-Sequenzen, die in Abhängigkeit ihrer Häufigkeit für die Extraktion auf einem Testkorpus selektiert wurden. Bei einem Schwellwert von 1 ist der Recall maximal, da alle generierten und zur Verfügung stehenden Patterns auch angewendet werden. Mit zunehmenden Schwellwert werden nur die am häufigsten generierten Patterns selektiert und getestet. Bei einem Schwellwert von 100 bleiben nur noch wenige Patterns übrig, die aufgrund der Häufigkeit ihrer Generierung die größte Wahrscheinlichkeit haben, eine passende Satzstelle zu finden. Neben einem einfachen Schwellwert können auch andere Auswahlkriterien, wie die Selektion anhand der individuellen Extraktionsergebnisse zu jedem Pattern, herangezogen werden, ohne damit jedoch für diesen Korpus weitere Verbesserungen zu erzielen. Tabelle 5.10 zeigt einige individuelle Patternresultate.

Verwendet man die generierten Pattern-Sequenzen als Mealy-Automaten zum Parsen mit einem Text, werden bei Verwendung aller Patterns eine Precision von 79% bei einem Recall von 18% und nach Zulassen beliebig langer Wordgaps ein Recall von 66% bei einer Precision von 29% erreicht. Bessere
5.5 Evaluation

Tabelle 5.6: Ausschnitt aus der Satzdatei des BCC mit dem zugehörigen Gold-Ausschnitt.

Ergebnisse werden mit Automaten erzielt, die von einem GA optimiert wurden. Die Abbildungen in 5.5 und 5.6 zeigen die Ergebnisse von evolvierten Automaten für die Fitnessmaße Precision und Recall. Sehr positiv ist der hohe Recall bei Verwendung sehr weniger Patterns. Ein Grund für die relativ geringe Precision der optimierten Patterns liegt in der Überanpassung (Overfitting) der gefundenen Modellparameter an die Trainingsmenge (Abb. 5.12).
Abbildung 5.4: Ergebnisse für das Pattern-Alignment auf dem BCC. Graphik (a) zeigt die Ergebnisse der Patterns, die aus dem eigenen Korpus generiert wurden. Für die Ergebnisse in Graphik (b) stammen die Patterns dagegen aus dem IEPA-Korpus und wurden ebenfalls auf dem BCC getestet. Die Graphiken (c) und (d) zeigen die entsprechenden Ergebnisse ohne Vergleich des Interaktionstyps bzw. des Interaktors einer PPI.

Durch Einschränkung der Transitionsstruktur auf Vorwärtsverknüpfungen wurde eine Anhebung und Annäherung der crossvalidierten Testkurve an die Trainingskurve erreicht. Verwendet man für diesen Korpus dagegen Automaten, die auf dem IEPA-Korpus trainiert wurden, werden noch bessere Ergebnisse erzielt (Abb. 5.5).
Abbildung 5.5: Ergebnisse für eine zunehmende Menge von Zustandsautomaten auf dem BCC. Die Automaten bestehen aus sechs Zuständen und sind auf das Fitnessmaß *Precision* optimiert. Die linken Graphiken zeigen Kurven für Automaten, die auf dem BCC optimiert wurden, mit (a) und ohne (c) Vergleich des Interaktors. Die Graphiken (b) und (d) zeigen die entsprechenden Ergebnisse von Automaten, die auf dem IEPA-Korpus optimiert und auf dem BCC validiert wurden. Graphik (b) zeigt, dass diese Automaten sehr präzise Vorhersagen treffen.

### 5.5.2 Der IEPA-Korpus

Der *Information Extraction Performance Assessment* (IEPA) Korpus von Berleant et al. enthält in den heruntergeladenen Dateien 489 Sätze mit annotierten Proteinen und Interaktionen, die aus insgesamt 303 Abstracts entnommen sind [Berleant et al., 2003]. Der Korpus wurde aus Microsoft Excel-Tabellen in ein tabulatorgetrenntes Format exportiert und dann in das bereits verwendete textbasierte Format des BCC übersetzt (Tab. 5.7). In Tabelle 5.5 sind weitere Korpus-Merkmale quantifiziert.
Abbildung 5.6: Ergebnisse für eine zunehmende Menge von Recall-optimierten Zustandsautomaten auf dem BCC. Die linken Graphiken zeigen Kurven für Automaten, die auf dem BCC optimiert wurden mit (a) und ohne (c) Vergleich des Interaktors. Die Graphiken (b) und (d) zeigen die entsprechenden Ergebnisse von Automaten, die auf dem IEPA-Korpus optimiert und auf dem BCC validiert wurden. Bereits nach vier Automaten wurden kaum noch weitere Interaktionen gefunden und die Evolution an dieser Stelle beendet.

In Tabelle 5.5 sind keine Angaben über die Anzahl der verschiedenen Stellungen von Proteinen und Interaktoren vorhanden, da dieser Korpus keine Tokenindizes der annotierten Namen enthält, ohne die kein automatisches Abzählen erfolgen konnte. Die Beispielsequenzen für die Eingabe des PGA sind in diesem Fall alle kompletten Sätze, die mindestens eine PPI enthalten. In den Abbildungen 5.7, 5.8 und 5.9 sind die Ergebnisse analog zum ersten Korpus präsentiert. Nach Transformierung der vom PGA generierten Sequen-

...
5.5 Evaluation

<table>
<thead>
<tr>
<th>Sätze</th>
<th>PPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>270 The/DT aggregated/JJ form/NN of/IN A/PTN beta/PTN 25-35/CD significantly/RB inhibited/VBD PIP2-PLC/PTN only/RB in/IN the/DTD presence/NN of/IN endogenous/JJ CaCl2/NN ./.</td>
<td>270</td>
</tr>
<tr>
<td>271 A/DTD neurotoxic/JJ fragment/NN of/IN amyloid/PTN ./, Abeta/NNP 25-35/CD ./, incubated/VBN in/IN the/DTD presence/NN of/IN endogenous/JJ Ca2+/NNS ./, increased/VBD significantly/RB the/DTD PI-PLC/PTN activity/NN of/IN normoxic/JJ brain/NN ./</td>
<td>271</td>
</tr>
<tr>
<td>272 In/IN its/PRP non/JJ aggregated/JJ form/NN ./, Abeta/PTN 25-35/PTN activates/VBZ PI-PLC/PTN but/CC in/IN the/DTD aggregated/JJ form/NN the/DTD enzymatic/JJ activity/NN decreased/VBD ./</td>
<td>272</td>
</tr>
</tbody>
</table>

Tabelle 5.7: Ausschnitt aus der Satzdatei des IEPA-Korpus mit dem zugehörigen Gold-Ausschnitt. Die Tokenindizes sind zu diesem Korpus nicht vorhanden und daher alle gleich 0.

Tatsächlich in Automaten, wird bei einem Wordgap von 0 eine Precision von 85% bei einem Recall von 16% erreicht. Nach Zulassen beliebig vieler Zwischenwörter steigt der Recall auf 49% bei einer Precision von 41%. In Abbildung 5.10 sind die Ergebnisse noch einmal zusammengefasst.

5.5.3 Die DIP

Abbildung 5.7: Ergebnisse für das Pattern-Alignment auf dem IEPA-Korpus. Graphik (a) zeigt die erzielten Ergebnisse von Patterns, die aus dem eigenen Korpus generiert wurden. Für die Ergebnisse in (b) stammen die Patterns dagegen aus dem BCC und wurden ebenfalls auf dem IEPA-Korpus getestet. Die Graphiken (c) und (d) zeigen die entsprechenden Ergebnisse ohne Vergleich des Interaktionstyps bzw. des Interaktors einer PPI.

Ein dritter Evaluationskorpus besteht aus einer Menge nachgewiesener Protein-Protein Interaktionen (PPI) im Hefe-Organismus mit den dazugehörigen SGD-IDs der Proteine und den PubMed-IDs der Artikel, in denen die Interaktionen beschrieben sind. Es wurden 165 zufällige PubMed-IDs aus der DIP ausgewählt, in denen mindestens eine und höchstens fünf Hefe-PPI referenziert sind. Verweisen viele PPI auf einen Artikel, so werden diese dort oft in graphischer oder tabellarischer Form dargestellt, woraus keine Information mit Textmining-Methoden extrahiert werden kann. In den selektierten
Abbildung 5.8: Ergebnisse für *Precision*-optimierte Zustandsautomaten auf dem IEPA-Korpus, mit (a) und ohne (c) Vergleich des Interaktionstyps. Die Graphiken (b) und (d) zeigen die entsprechenden Ergebnisse von Patterns, die aus dem BCC generiert und auf IEPA-Korpus validiert wurden.

Abbildung 5.9: Ergebnisse mit Recall-optimierten Zustandsautomaten auf dem IEPA-Korpus, mit (a) und ohne (c) Vergleich des Interaktionstyps. Die Graphiken (b) und (d) zeigen die entsprechenden Ergebnisse von Patterns, die aus dem BCC generiert und auf IEPA-Korpus validiert wurden.

celten Protein-Tagger markiert. Tabelle 5.8 zeigt einen Ausschnitt aus der Liste der 297 selektierten Interaktionen. Zu den SGD-IDs liegt eine Synonymliste mit den in der Literatur oft gebrauchten Bezeichnern vor, die für eine Normalisierung der Namen herangezogen wird (Tab. 5.9). Tabelle 5.5 enthält auch zu diesem Korpus einige Merkmale. Viele sind nicht aufgelistet, da ihre Bestimmung eine genaue Untersuchung der mehr als 100000 Sätze benötigt hätte. Unter diesem Gesichtspunkt müssen auch die Ergebnisse der Evaluation betrachtet werden. Der gemessene Recall ist relativ zu der Gesamtmenge der 297 PPI zu bewerten, deren Existenz in diesen Artikeln gesichert ist. Die
5.6 Fehleranalyse

<table>
<thead>
<tr>
<th>DIPEdgeID</th>
<th>SGD-ID 1</th>
<th>SGD-ID 2</th>
<th>PubMedID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2444</td>
<td>S0001297</td>
<td>S0001595</td>
<td>7608180</td>
</tr>
<tr>
<td>2445</td>
<td>S0005587</td>
<td>S0001595</td>
<td>7608180</td>
</tr>
<tr>
<td>2446</td>
<td>S0002987</td>
<td>S0001595</td>
<td>7608180</td>
</tr>
<tr>
<td>2447</td>
<td>S0005565</td>
<td>S0001595</td>
<td>7608180</td>
</tr>
<tr>
<td>2797</td>
<td>S0003041</td>
<td>S0005704</td>
<td>10207049</td>
</tr>
<tr>
<td>2960</td>
<td>S0002719</td>
<td>S0003995</td>
<td>8171014</td>
</tr>
<tr>
<td>3210</td>
<td>S0001405</td>
<td>S0000973</td>
<td>8171014</td>
</tr>
<tr>
<td>10404</td>
<td>S0000929</td>
<td>S0005370</td>
<td>10790377</td>
</tr>
<tr>
<td>10405</td>
<td>S0002172</td>
<td>S0005370</td>
<td>10790377</td>
</tr>
<tr>
<td>17764</td>
<td>S0000872</td>
<td>S0000872</td>
<td>11893751</td>
</tr>
</tbody>
</table>

Tabelle 5.8: Ausschnitt aus der Liste der 297 selektierten Hefe-Interaktionen aus der DIP.

*Precision* wurde aus Gründen der großen Menge an extrahierten Interaktionen nicht gemessen, da eine manuelle Überprüfung aller gefundenen PPI zu zeitaufwändig wäre.

Bei einem Alignment mit allen generierten Patterns werden 60% der 297 gesuchten Hefe-PPI gefunden. Die $10 \times 4$ Recall-optimierten Automaten finden im Schnitt 65% der 297 Hefe-Interaktionen aus der DIP und erzielten damit das bisher beste Ergebnis. Zu 63% der nicht gefundenen PPI lag nur der Abstract vor. Ein Volltext fehlt, wenn über PubMed kein Link angeboten wird oder das PDF nicht kostenfrei heruntergeladen werden kann. Die Wiedererkennungsrate von PPI, zu denen ein Volltext vorhanden ist, liegt bei 82% (167/202). In Abbildung 5.10 sind noch einmal alle Ergebnisse zusammengefasst.

5.6 Fehleranalyse

In diesem Kapitel wird genauer auf die Vor- und Nachteile der untersuchten Methoden und ihre typischen Fehlerquellen eingegangen. Die Analyse wird vorwiegend am BCC durchgeführt, da für diesen die meisten Sätze zur Verfügung stehen, vor allem auch die genauen Tokenindizes der beteiligten, interagierenden Entitäten. Der Anteil an interaktionsenthaltenden Sätzen zu den übrigen ist im BCC realistischer als im IEPA. Die Evaluationsergebnisse
SGD-ID | Synonyms
--- | ---
S0002719 | YDR311W TFB1
S0001405 | YIL143C SSL2 RAD25 LOM3
S0001297 | YIL035C CKA1
S0005587 | YOR061W CKA2 YOR29-12
S0002987 | YGL019W CKB1
S0005565 | YOR039W CKB2
S0000929 | YER127W LCP5
S0002172 | YDL014W NOP1 LOT3
S0003041 | YGL073W HSF1 MAS3 EXA3
S0000872 | YER070W RNR1 SDS12 RIR1 CRT7
S0003995 | YLR005W SSL1
S0000973 | YER171W RAD3 REM1
S0001595 | YKL112W ABF1 SBF1 REB2 OBF1 BAF1
S0005370 | YOL010W RCL1
S0005704 | YOR178C GAC1


fielen auch für den BCC insgesamt besser aus. Es soll nun untersucht werden, welche Fehler bei jedem Verfahren auftreten und worin ihre Ursachen liegen.

5.6.1 Pattern-Alignment

Ein Alignment mit wenigen Patterns, die mit der größten Häufigkeit aus den Trainingsbeispielen generiert wurden, erzielt für die Precision sehr gute Ergebnisse von über 90% und 83% auf den Satzkorpora Korpora BCC und IEPA. Zunächst werden die Fehler untersucht, die einer Pattern-Menge mit hoher Precision dennoch unterlaufen. Nach einer zehnfachen Durchführung einer PPI-Extraktion auf dem gesamten BCC blieben drei Fehler übrig, die in allen zehn Versuchen auftraten. Der erste Fehler unterlief bei der Vorhersage einer Interaktion zwischen NOT4 und not1 am Ende des Satzes:

NOT4/PTN interacts/NN with/IN NOT1/PTN and/CC NOT3/PTN
5.6 Fehleranalyse

Abbildung 5.10: Zusammenfassung der besten erreichten Ergebnisse zu beiden Verfahren. Für die dargestellten Ergebnisse auf den Korpora BCC und IEPA wurden die Patterns jeweils aus dem eigenen Korpus generiert. Für die DIP wurden die Patterns aus dem BCC verwendet.

An dieser Stelle ist keine Interaktion zwischen diesen Proteinen beschrieben, sondern eine Auswirkung der Exprimierung von NOT4 auf Mutationen von not1. Dieser Fehler tritt unweigerlich auf, wenn der Text an dieser Stelle mit der Pattern-Sequenz 'PTN vbint PTN' aligniert wird. Die Wörter außerhalb der Sequenz werden für die Entscheidung nicht betrachtet. Der zweite Fehler trat im folgenden Satz auf:

An/DT endogenous/JJ mammalian/JJ regulator/NN of/IN this/DT process/NN ,/ named/VB Usurpin/PTN ,/ has/VB been/VB identified/VB (/(/ aliases/NN for/IN Usurpin/PTN include/VB CASH/PTN ,/ Casper/PTN ,/ CLARP/PTN ,/ FLAME-1/PTN ,/ FLIP/PTN ,/ I-FLICE/PTN and/CC MRIT/PTN )) ./.

Die extrahierte PPI zwischen Usurpin, CASH und include innerhalb der Klammer ist falsch, da hier nur eine Aufzählung von Synonymen ge-

Der dritte Fehler unterlief in dem folgenden Satz:

\[\text{The results of these studies demonstrate that the rat HDL receptor SR-BI promoter contains two sterol response elements (\(pSRE\) and \(dSRE\)) through which SREBP-1a can bind and activate transcription of this gene.}\]

Es wurde eine PPI zwischen \textit{rat HDL receptor SR-BI promoter}, \textit{sterol response elements} und \textit{contains} extrahiert, die nicht annotiert und daher falsch ist. Die Ursache für die Extraktion liegt, wie im ersten Fall, in der Ähnlichkeit zum häufigsten Pattern 'PTN VBINT PTN', nur das der Fehler in diesem Fall nicht gravierend, unter gewissen Gesichtspunkten gar kein Fehler ist.

Eine weitere Möglichkeit Patterns auszuwählen besteht darin, die Wahl nach ihren individuellen Extraktionsergebnissen abhängig zu machen. Statt alle Patterns mit einem hohen Support zu selektieren, werden z.B. nur all jene gewählt, die auf dem Trainingskorpus eine Precision von mehr als 95% erreicht haben oder nach anderen Kriterien, wie einen hohen Recall. Eine Evaluation mit Patterns, die aufgrund ihrer guten Precision-Werte selektiert wurden, ergab jedoch bei der Validation keine Verbesserungen, eher fielen die Ergebnisse weiter ab, was darauf hindeutet, das auch Patterns mit schlechteren Trainingsergebnissen auf dem Testkorpus wieder besser abschneiden. In Tabelle 5.10 wird diese Vermutung durch individuell gemessene Ergebnisse bestätigt. Um stattdessen den Recall weiter zu erhöhen, kann zunächst die Anzahl der generierten Patterns durch Anwendung von Alignment-Funktionen, die gleichzeitig mehrere Matches zwischen zwei Se-
5.6 Fehleranalyse

<table>
<thead>
<tr>
<th>Support</th>
<th>Pattern-Sequenz</th>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>487</td>
<td>PTN VBINT PTN</td>
<td>0.83</td>
<td>1.0</td>
</tr>
<tr>
<td>97</td>
<td>PTN VBINT IN PTN</td>
<td>0.94</td>
<td>0.08</td>
</tr>
<tr>
<td>73</td>
<td>PTN VBINT DT PTN</td>
<td>0.86</td>
<td>1.0</td>
</tr>
<tr>
<td>54</td>
<td>PTN NNINT IN PTN</td>
<td>0.50</td>
<td>0.02</td>
</tr>
<tr>
<td>47</td>
<td>PTN NNINT PTN</td>
<td>0.66</td>
<td>1.0</td>
</tr>
<tr>
<td>31</td>
<td>PTN VBINT IN DT PTN</td>
<td>0.81</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Tabelle 5.10: Individuelle Ergebnisse nach einem Alignment einiger Patterns auf einer Trainings- und Testmenge.

quenzen berechnen (Repeat-Matches), erhöht werden. Nach der Anwendung eines Alignment-Algorithmus, der überhängende Enden nicht bestraft (End-Space-Free), erhöhte sich der durchschnittliche Recall um etwa 10%. Abbildung 5.11 zeigt die entsprechenden crossvalidierten Kurven für die Korpora BCC und IEPA. Ein Nachteil dieser Variante war jedoch die hohe Zahl an optimalen Alignments, die alle zu berücksichtigen sind, weshalb die Zeit- und Speicheranforderungen mit diesem Algorithmus noch mehr zunahmen.

Nach zehnfacher Durchführung der Extraktion mit jeweils allen Patterns aus einer Trainingsmenge blieben 58 PPI im BCC unentdeckt. Einige Beispiele seien hierfür angeführt, wie die nicht gefundenen Interaktionen in dem Satz:

\[ Sp1/PTN \text{ and/CC two/CD } Sp3/PTN \text{ isoforms/PTN were detected/VB as/IN the/DT primary/JJ cellular/JJ constituents/NN of/IN DNA-protein/NN complexes/NN formed/VB with/IN the/DT NF-kappaB-proximal/PTN site/PTN ./}. \]

Die drei Proteine \(Sp1\), \(Sp3\) isoforms und \(NF\)-kappaB-proximal site formen eines Complex, der in der Musterlösung dadurch beschrieben ist, dass alle drei Proteine untereinander in bidirektionaler Beziehung stehen. Der Interaktionstyp ist in dem Fall complex(es). Von den drei PPI wurde nur eine zwischen \(Sp3\) und \(NF\)-kappaB entdeckt.

Abschließend wird noch folgendes Beispiel angeführt:

\[ The/DT localization/NN of/IN ZAP-70/PTN to/TO the/DT cell \]
Abbildung 5.11: Ergebnisse für ein Pattern-Alignment auf dem BCC (a) und dem IEPA-Korpus (b) mit einem End-Space-Free-Algorithmus. Im Gegensatz zur Abbildung 5.7 schneiden sich die Precision- und Recall-Kurven nun auch auf dem IEPA-Korpus.

cortex/NN is/VB ./, therefore/RB ./, regulated/VB by/IN the/DT activity/NN of/IN SRC-family/PTN kinases/PTN ./, independently/RB of/IN their/PRP ability/NN to/TO phosphorylate/VB immunoreceptor/NN tyrosine-based/JJ activation/NN motifs/NN of/IN the/DT TCR/PTN ./.


5.6.2 Zustandsautomaten

Die angesprochenen Fehler beim Alignment von Patterns mit hoher Präzision treten auch bei den optimierten Mealy-Automaten auf. Nach dem Testen von zehn verschiedenen Patterns auf dem BCC blieben sechs FPs übrig, die in allen Versuchen vorkamen, darunter der Fehler in dem Satz:

Chick/PTN brain/PTN actin/PTN depolymerizing/PTN factor/PTN (/(/ADF/PTN /)/) is/VB a/DT 19-kDa/CD protein/NN that/IN severs/NN actin/PTN filaments/PTN and/CC binds/VB actin/PTN monomers/PTN ./.
In diesem Satz wird eine Bindung zwischen den Proteinen actin filaments und actin monomers erkannt, ein Fehler der auch bei einem Alignment mit der Patternsequenz 'PTN VBINT PTN' auftreten kann.

In dem nächsten Satz wurde eine falsche Bind-Interaktion zwischen C / EBP alpha and beta und CREB extrahiert:

This construct, termed pDeltaCREC/PTN //PTN EBP/PTN , binds/ VB C/PTN //PTN EBPalpha/PTN and/PTN beta/PTN but/CC not/RB CREB/PTN ./, yet/ RB it/ PRP confers/ VB a/ DT nearly/ RB complete/ JJ glucocorticoid/ JJ response/ NN when/ WRB transiently/ RB transfected/ VB into/ IN H4IIE/ NN rat/ NN hepatoma/ NN cells/ NN ./.


Als letzter Fehler wird die falsche PPI zwischen PKC und TPO angeführt:

TPO/PTN by/ IN itself/ PRP did/ VB not/ RB activate/ VBP ERK1/ PTN ./, ERK2/ PTN and/ CC protein/ PTN kinase/ PTN C/PTN (/( PKC/PTN )/) ./, whereas/ IN TPO/ PTN directly/ RB enhanced/ VB the/ DT PKC-dependent/ JJ activation/ NN of/ IN ERKs/ PTN induced/ VBN by/ IN other/ JJ agonists/ NN including/ VB thrombin/ PTN and/ CC phorbol/ JJ esters/ NN ./, without/ IN affecting/ VB the/ DT PKC/ PTN activation/ NN by/ IN those/ DT agonists/ NN ./.

Die extrahierte Interaktion erstreckt sich über zwei Teilsätze und ergibt so keinen Sinn. Insgesamt treten beim Parsen mit Zustandsautomaten die gleichen oder ähnlichen Fehler wie bei einem Pattern-Alignment auf. Eine
Anwendung: Protein-Protein-Interaktionen

Ursache für die etwas höhere Anzahl an Fehlern ist das konsequente Parsen des Textes, beginnend bei jedem Wort, wohingegen bei einem Alignement nur das optimale interpretiert wird.

Beim Vergleich der Lernkurven für eine zunehmende Menge von GA-optimierten Patterns auf den Trainings- und Testdatenmengen des BCC und IEPA-Korpus zeigt sich ein Overfitting hinsichtlich der Trainingsmenge. Durch Einschränkung der Transitionsstruktur auf vorwärtsverknüpfte Zustandsübergänge und damit einhergehender Modellbeschränkung wird eine Annäherung der Lernkurven erreicht (Abb. 5.12).

Nach Anwendung der optimierten IEPA-Automaten auf den Sätzen des BCC erzielten diese eine Precision von über 90%. Der Anteil an positiven zu negativen Sätzen ist im BCC zu Gunsten der negativen wesentlich höher, was eher einer Verteilung entspricht, wie sie in einem realen Text vorkommt. Die Vorteile des Mealy-Ansatzes liegen in einem hohen Recall bei wenigen und einfachen Automaten, mit denen ein Text sehr schnell ge-Parst und interpretiert werden kann. Regelt man die Pattern-Menge für ein Alignment mit hohem Recall über den Schwellwert des PGA, werden immer alle oder sehr viele Patterns selektiert, mit denen die Extraktion wesentlich zeitaufwändiger ist (siehe Kapitel 5.7). Nach der zehnfachen Extraktion mit den jeweils vier besten Recall-Automaten blieben elf FNs übrig, die von keinem Pattern gefunden wurden, darunter die Interaktion in dem Satz:

This/DT cluster/NN consisted/VB of/IN four/CD apparently/RB unrelated/JJ ESTs/NN and/CC two/CD genes/NN ,/, pregnancy-associated/PTN plasma/PTN protein-A/PTN ((/ PAPP-A/PTN )/) and/CC a/DT novel/NN gene/NN ((/ tentatively/RB named/VB EST-YD1/PTN )/) ./. 

Diese beiden Proteine bilden einen Cluster oder Complex und müssen als solche gefunden werden. Eine weitere Interaktion ist in dem folgenden Satz beschrieben:

While/IN this/DT Saccharomyces/NN cerevisiae/JJ SIN4/PTN gene/PTN product/PTN is/VB a/DT component/NN of/IN a/DT mediator/NN complex/NN associated/VB with/IN

Diese Interaktion wurde vielleicht aufgrund der beiden Interakoren mediator oder complex vor dem Wort associated nicht gefunden, die stattdessen der PPI als Interaktionstyp zugeordnet wurden. Die folgende beschriebene PPI zwischen Mod und homeotic gene vom Typ regulation wurde wahrschein-
lichen aufgrund der Länge des Satzes nicht entdeckt. Da für das Protein *Mod* auch ein Synonym *E ( var ) 3-93D* angegeben ist, hätte auch eine gefundene PPI mit diesem Interaktionspartner gegolten:

\[ \text{Mod/PTN (}/\text{PTN mdg4/PTN})/\text{PTN ./, also/RB known/VB as/IN E/PTN (}/\text{PTN var/PTN})/\text{PTN 3-93D/PTN ./, is/VB involved/VB in/IN a/DT variety/NN of/IN processes/NN ./, such/JJ as/IN gene/NN silencing/NN in/IN position/NN effect/NN variegation/NN } (\text{PEV/NN}) \text{)/, the/DT control/NN of/IN gypsy/PTN insulator/PTN sequences/PTN ./, regulation/NN of/IN homeotic/PTN gene/PTN expression/NN ./, and/CC programmed/JJ cell/NN death/NN ./. } \]

Die folgenden drei, in ihren Sätzen unterstrichenen Interaktion wurden ebenfalls nicht gefunden. Diese zeichnen sich alle durch ein ähnliches Muster in ihrer Beschreibung aus: 'PTN .. no/not .. PTN .. but .. INT .. PTN'.

1. Although/IN linker/NN regions/NN in/IN transcription/NN factors/NN are/VB known/VB to/TO modulate/VB DNA/NN binding/JJ specificity/NN ./, our/PRP studies/NN suggest/VB that/IN the/DT human/PTN HSF1/PTN linker/NN plays/VB no/DT role/NN in/IN determining/VB HSF1/PTN binding/JJ preferences/NN in/IN vivo/NN but/CC is/VB a/DT critical/JJ determinant/NN in/IN regulating/VB the/DT HSF1/PTN monomer-trimer/PTN equilibrium/NN ./. 

2. However/RB ./, cotransfection/NN studies/NN indicate/VB that/IN RVR/PTN does/VB not/RB activate/VB transcription/NN when/WRB this/DT hormone/PTN response/PTN element/PTN is/VB linked/VB to/TO a/DT reporter/NN gene/NN but/CC rather/RB acts/VB as/IN a/DT potent/JJ competitive/JJ repressor/NN of/IN ROR/PTN alpha/PTN function/NN ./. 

3. Dominant/NN negative/JJ Sos/PTN did/VB not/RB affect/VB carbachol/JJ stimulation/NN of/IN HA-ERK2/PTN but/CC inhibited/VB the/DT stimulatory/JJ effect/NN of/IN EGF/PTN by/IN 60%/CD ./.
Die übrigen fünf nicht gefundenen PPI seien zum Abschluß noch in ihren Sätzen aufgelistet und unterstrichen.

1. *The* specific interaction between *a* defined structural element of *the* human *immunodeficiency* virus mRNA *(RRE*, the *Rev* response element) and *the* virus-encoded protein *Rev* has been implicated in *the* regulation of *the* export of unspliced or singly spliced mRNA from *the* nucleus to *the* cytoplasm.

2. In flies, *the* dShc protein physically associates with *the* activated Drosophila epidermal growth factor receptor homolog *(DER)* and is inducibly phosphorylated on tyrosine by *DER*.

3. *The* results of these studies demonstrate that *the* rat HDL receptor SR-BI promoter contains two sterol response elements *(pSRE and dSRE)* through which SREBP-1a can bind and activate transcription of this gene.

4. Gel filtration, sedimentation velocity, and immunoprecipitation experiments revealed that *beta4* is a component of a multisubunit complex *(AP-4)* that also contains the sigma4 polypeptide and two additional adaptor subunit homologs named mu4 and epsilon.
5. RPM/PTN // PTN RGL3/PTN resembled/VB AF-6/PTN and/CC
Nore1/PTN in/IN interacting/VB strongly/RB with/IN
constitutively/RB active/JJ M-Ras/PTN and/CC
p21/PTN Ras/PTN ./.

5.7 Zeitanalyse

Für einen abschließenden Vergleich werden die Laufzeiten der implementierten und eingesetzten Extraktionskomponenten zu jeder Methode gemessen. Die Zeitmessung beginnt vor dem übergeben der Sätze an die process-Methode der Komponente und endet nach Rückkehr aller Aufrufe, d.h. die Vorverarbeitung und Annotation ist erfolgt und nur die Dauer des letzten Extraktionschrittes zur Bestimmung der Relationen wird gemessen.

Das Parsen eines Textes $t$ mit einem Pattern $p$ aus einem deterministischen Mealy-Automaten wurde naiv mit der Laufzeit $O(|t| \times |p|)$ implementiert. Tabelle 5.11 zeigt die gemessenen Zeiten für verschiedene Textgrößen.

<table>
<thead>
<tr>
<th>Mealy-FSAs</th>
<th>Sätze</th>
<th>200</th>
<th>400</th>
<th>600</th>
<th>800</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td></td>
<td>0.08 s</td>
<td>0.13 s</td>
<td>0.17 s</td>
<td>0.22 s</td>
<td>0.26 s</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>0.11 s</td>
<td>0.19 s</td>
<td>0.26 s</td>
<td>0.34 s</td>
<td>0.41 s</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>0.17 s</td>
<td>0.34 s</td>
<td>0.47 s</td>
<td>0.63 s</td>
<td>0.75 s</td>
</tr>
</tbody>
</table>

Tabelle 5.11: Extraktionszeiten der Komponente mit Patterns aus Mealy-Automaten.
5.7 Zeitanalyse

Tabelle 5.12: Extraktionszeiten der Komponente für das Pattern-Alignment.

<table>
<thead>
<tr>
<th>Patterns</th>
<th>Sätze</th>
<th>200</th>
<th>400</th>
<th>600</th>
<th>800</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.31 s</td>
<td>0.63 s</td>
<td>0.94 s</td>
<td>1.32 s</td>
<td>1.59 s</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.54 s</td>
<td>1.10 s</td>
<td>1.61 s</td>
<td>2.21 s</td>
<td>2.64 s</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>1.31 s</td>
<td>2.68 s</td>
<td>4.01 s</td>
<td>5.74 s</td>
<td>6.84 s</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>3.24 s</td>
<td>6.43 s</td>
<td>9.33 s</td>
<td>12.6 s</td>
<td>15.1 s</td>
<td></td>
</tr>
</tbody>
</table>

Ergebnissen gelangen. Zu beachten ist, dass Sätze, die keine Proteine oder Interaktoren enthalten, nicht geparst werden.
Kapitel 6

Zusammenfassung

Diese Arbeit stellte sich dem Problem der Informations-Extraktion (IE) aus freien, natürlichsprachlichen Texten. Es wurde ein formales Konzept eines Textes eingeführt und die darin enthaltene Information als benannte Entitäten und Relationen zwischen diesen definiert (Kapitel 2). Für die Bestimmung der Relationen in einem Text wurden verschiedene Ansätze in Form eines binären Klassifikators kurz vorgestellt (Kapitel 3) und zwei Methoden, Sequenz-Alignment und endliche Automaten, in einer Anwendung zur Extraktion von PPI aus wissenschaftlichen Artikeln genauer untersucht (Kapitel 5).


Ein Problem des Mealy-Ansatzes ist die Tendenz zum Overfitting beim Lernen Precision-optimierter Automaten, was sich beim Vergleich der Lernkurven auf den Trainings- und Testmengen der beiden Korpora BCC und IE-
PA zeigt (Abb. 5.12). Um dem entgegenzuwirken, wurde die Modellstruktur, hier Transitionstruktur, vereinfacht. Die Verwendung von Automaten, deren Zustände nur Transitionen auf sich selbst und auf Zustände mit größerem Index erlauben, zeigen bereits bessere Ergebnisse. Mit dieser Einschränkung sind nicht mehr alle regulären Sprachen abgedeckt, die endlichen bleiben es aber dennoch, was für die Analyse endlich langer Sätze, die in der Literatur ausschließlich vorkommen, genügt.

Während der Evaluation wurde immer das gleiche Alphabet verwendet (Tab. 5.1). Die Auswirkung verschiedener POS-Tags auf die Extraktionsergebnisse ist noch zu untersuchen. Durch die Einbeziehung von Chunking- oder SPO-Informationen\(^1\) können die Sätze vereinfacht oder mit zusätzlichen Annotationen aufgewertet werden. Hierfür sind weitere Crossvalidatio- nen durchzuführen. Die Bestimmung der Interaktoren zu den PPI wurde anhand eines vorher festgelegten Wörterbuchs sehr einfach vollzogen. Wie die Evaluationsergebnisse von der Menge der Interaktoren abhängen wurde nicht weiter untersucht.

Ein positives Ergebnis dieser Arbeit sind zwei unterschiedliche, implementierte Verfahren zur Extraktion von PPI, die durch einfache Anpassung des Alphabetes und der Substitutionsmatrix innerhalb eines Java-Frameworks auch für andere Anwendungen eingesetzt werden können. Die generierten oder evolvierten Patterns für die PPI-Extraktion stehen nach kurzer Zeit, d.h. innerhalb einer Minute zur Verfügung, zeigen gute Ergebnisse und können mit anderen, auch manuell konfigurierten, Verfahren konkurrieren. Ebenfalls positiv zu bemerken sind die guten Ergebnisse von Patterns, die auf einem Korpus gelernt und auf einem anderen validiert wurden, was auf eine gewisse Robustheit dieser Patterns hindeutet.


\(^1\)Subjekt-Prädikat-Objekt


Kapitel 7

Ausblick


In den bisher angesprochenen Anwendungen haben die Relationen eine fixe Stelligkeit, wohingegen auch eine Extraktion von Relationen über beliebig viele Entitäten denkbar wäre. Zum Beispiel wäre das Zulassen von mehreren Interaktoren zu einer PPI eine sinnvolle Erweiterung, da sich diese Information auch in mehreren Wörtern ausdrücken kann. Die PPI sind manchmal mehrdeutig beschrieben, was die Festlegung eines einzigen Typenwortes erschwert und nur durch einen Kompromiss erreicht wird. Um die Evaluation etwas aufzulockern, wird dann beim Vergleich der Interaktoren zu zwei PPI nur geprüft, ob die Schnittmenge leer ist oder nicht.

Die Bestimmung zusammengehöriger Wörter zu einem Proteinnamen, wie sie in dem Post-Processing Schritt des Protein-Taggers vorgenommen wird,

Anhang A

Sequenz-Diagramme
Literaturverzeichnis


Tabellenverzeichnis

3.1 Beispiel für einen vollständig POS-annotierten Satz. .......... 13
3.2 Alignment zwischen zwei POS-Sequenzen. ................. 15

5.1 Alphabet für die Anwendung: Protein-Protein-Interaktionen . 32
5.2 Liste der Wortstämme zu allen bekannten Interaktoren. .... 33
5.3 Ausschnitt aus der Substitutionsmatrix für POS- und NER-Tags,
die bei der Extraktion von PPI verwendet werden. .......... 34
5.4 Generierte Pattern-Sequenzen aus dem BioCreative Corpus. . 35
5.5 Ausgewählte Merkmale der drei Evaluationskorpora. ........ 48
5.6 Ausschnitt aus der Satzdatei des BCC mit dem zugehörigen Gold-
Ausschnitt. ............................................. 49
5.7 Ausschnitt aus der Satzdatei des IEPA-Korpus mit dem zu-
gehörrigen Gold-Ausschnitt. ................................ 53
5.8 Ausschnitt aus der Liste der 297 selektierten Hefe-Interaktionen
aus der DIP. ............................................. 57
5.9 Ausschnitt aus der Synonymdatei zu den Hefe-Proteinen mit
zugehöriger SGD-ID. ..................................... 58
5.10 Individuelle Ergebnisse nach einem Alignment einiger Patterns auf
einer Trainings- und Testmenge. .............................. 61
5.11 Extraktionszeiten der Komponente mit Patterns aus Mealy-Auto-
matten. .................................................. 68
5.12 Extraktionszeiten der Komponente für das Pattern-Alignment. . 69
Abbildungverzeichnis

<table>
<thead>
<tr>
<th>Seite</th>
<th>Titel</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Ein Pattern zur Erkennung von Protein-Protein-Interaktionen.</td>
</tr>
<tr>
<td>3.2</td>
<td>Mealy-Automat für annotierte Sequenzen in einem Text.</td>
</tr>
<tr>
<td>3.3</td>
<td>Drei Pattern-Modelle.</td>
</tr>
<tr>
<td>3.4</td>
<td>Workflow zum Erlernen eines Extraktionsmodells.</td>
</tr>
<tr>
<td>3.5</td>
<td>Der Pattern-Generating Algorithmus.</td>
</tr>
<tr>
<td>4.1</td>
<td>Beispiel einer Textmining-Pipeline mit vier Komponenten.</td>
</tr>
<tr>
<td>5.1</td>
<td>Beispiel zweier Sätze, die Interaktionen beschreiben.</td>
</tr>
<tr>
<td>5.2</td>
<td>Ausschnitt aus einer Patternsequenz mit einer Transition für ein Protein-Tag.</td>
</tr>
<tr>
<td>5.3</td>
<td>Komponente zur Extraktion von PPI.</td>
</tr>
<tr>
<td>5.4</td>
<td>Ergebnisse für das Pattern-Alignment auf dem BCC.</td>
</tr>
<tr>
<td>5.5</td>
<td>Ergebnisse Precision-optimierter Zustandsautomaten auf dem BCC.</td>
</tr>
<tr>
<td>5.6</td>
<td>Ergebnisse Recall-optimierter Zustandsautomaten auf dem BCC.</td>
</tr>
<tr>
<td>5.7</td>
<td>Ergebnisse für das Pattern-Alignment auf dem IEPA-Korpus.</td>
</tr>
<tr>
<td>5.8</td>
<td>Ergebnisse Precision-optimierter Zustandsautomaten auf dem IEPA-Korpus.</td>
</tr>
<tr>
<td>5.9</td>
<td>Ergebnisse Recall-optimierter Zustandsautomaten auf dem IEPA-Korpus.</td>
</tr>
<tr>
<td>5.10</td>
<td>Zusammenfassung der besten erreichten Ergebnisse zu beiden Verfahren.</td>
</tr>
<tr>
<td>5.11</td>
<td>Ergebnisse für ein Pattern-Alignment mit einem End-Space-Free-Algorithmus.</td>
</tr>
</tbody>
</table>
5.12 Lernkurven auf den Trainings- und Testmengen des BCC und IEPA-Korpus.

A.1 Sequenz-Diagramme zu den Klassen BCEvaluation und DIPEvaluation.