**Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH**

Document
D-97-04

Entwurf einer Patternbeschreibungssprache für die Informationsextraktion in der Dokumentanalyse

Claudia Wenzel, Markus Junker

September 1997

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Entwurf einer Patternbeschreibungssprache für die Informationsextraktion in der Dokumentanalyse

Claudia Wenzel, Markus Junker

DFKI-D-97-04

This work has been supported by a grant from The Federal Ministry of Education, Science, Research, and Technology (FKZ ITWM-9702).

ISSN 0946-0098

Entwurf einer Patternbeschreibungssprache für die Informationsextraktion in der Dokumentanalyse

Claudia Wenzel, Markus Junker

September 1997

Inhaltsverzeichnis

Zusammenfassung

Dokumentanalyse befaßt sich mit der Extraktion von relevanten Informationen aus Dokumenten, die in Papierform vorliegen. Um die gewünschten Informationen in einem Text zu finden, können verschiedene Techniken angewendet werden. Sie reichen von einfachen Suchverfahren hin zum Versuch des vollständigen Parsens eines Textes. Häufig stammen diese Techniken aus dem Bereich der NLP, wo sie zur Verarbeitung von elektronischen Texten eingesetzt werden.

Unabhängig von der eingesetzten Technik benötigt man jedoch immer eine Sprache, mit der die Syntax und die Semantik der gesuchten Informationen beschrieben werden können. In diesem Dokument wird eine solche Sprache vorgestellt, die insbesondere den Erfordernissen der Dokumentanalyse Rechnung trägt, aber allerdings auch für die Verarbeitung elektronischer Texte genutzt werden kann. Derzeit wird die Sprache zur Informationsextraktion in und zur Klassifikation von deutschen Geschäftsbriefen eingesetzt.

1.0 Einführung

Systeme, die Dokumentanalyse und Dokumentverstehen leisten, haben nicht nur das Ziel, Zeichen- und Dokumentstrukturerkennung von gedruckten Texten durchzuführen, sondern es geht auch darum, die inhaltlich relevanten Teile des Textes zu identifizieren und zu extrahieren. Die inhaltliche Trennung von Dokumentanalyse und Dokumentverstehen (DAU) ist in

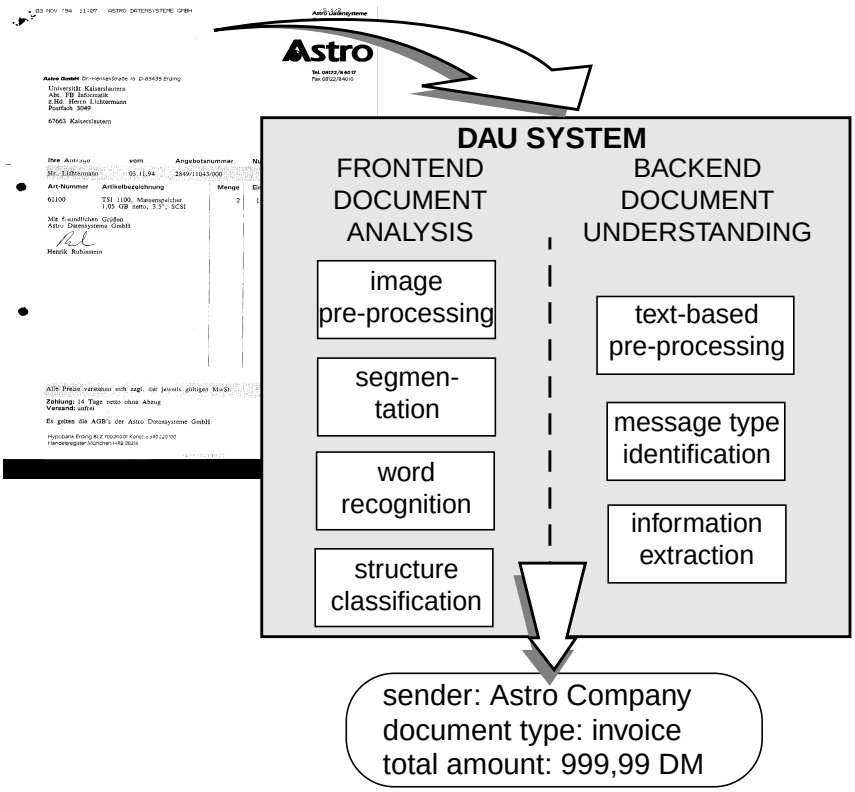

ABBILDUNG 1: Überblick über ein Gesamtsystem für Dokumentanalyse und Dokumentverstehen

Abbildung 1 dargestellt.

Der vordere Teil der Dokumentanalyse segmentiert ein Dokument in Zeichen, Wörter, Zeilen und Blöcke, erkennt den Text im Dokument und ordnet Teilen des Textes inhaltliche Bezeichner zu (z.B. Empfänger im Brief links oben). Der hintere Teil des Dokumentverstehens befaßt sich mit der inhaltlichen Erschließung des Textes auf der Basis der verfügbaren Ergebnisse der Dokumentanalyse. Eine Aufgabe besteht in der Informationsextraktion, die im Gegensatz zur Aufgabe des Textverstehens nicht die vollständige Semantik eines natürlichsprachlichen Textes ergründet, sondern gezielt die relevanten Informationen aus Texten herauszieht. Diese eher oberflächliche Form der Bearbeitung hat den Vorteil, daß man einfachere und schnellere Verfahren verwenden kann.

Dazu benötigt man eine Beschreibungssprache, die es ermöglicht, die syntaktische Erscheinungsform der zu extrahierenden Informationen zu beschreiben und sie mit einer Semantik zu verbinden. Wir nennen eine solche Sprache eine Patternbeschreibungssprache, wobei ein (Text-)Pattern[1] eine generische Beschreibung für eine im Dokument auftretende Patterninstanz ist.

1. Ein (Text-)Pattern (auch Phrase genannt) besteht aus einer Menge generischer Wortspezifikationen, die in einer Sequenz angeordnet sind, die die Reihenfolge von Instanzen dieser Spezifikationen im Text widerspiegeln sollen.

Je nach Aufgabe können die gefundenen Patterninstanzen unterschiedlich weiterverarbeitet werden: In der Textklassifikation ist z.B. nur das Auftreten und nicht der konkrete Inhalt der Patterninstanz von Bedeutung, wohingegen man bei der tatsächlichen Extraktion gerade den konkreten Inhalt einer generisch definierten Struktur benötigt.

Die Abarbeitung der Pattern wird hier außer Acht gelassen, sie kann z.B. mithilfe eines Pattern-Matchers [1] geschehen.

In diesem Dokument soll der Entwurf einer Beschreibungssprache vorgestellt werden, die insbesondere

- für unterschiedliche Aufgaben eingesetzt werden kann (z.B. Klassifikation, Info-Extraktion, Lernen für die Textklassifikation, Tabellenanalyse, Parsing),
- für verschiedene Domänen verwendbar ist,
- in ihrer Mächtigkeit sowohl gescannte Dokumente als auch ASCII-Texte beschreiben kann,
- benutzerfreundlich ist und
- in eine globalere (Domänen-) Dokumentrepräsentationssprache eingebunden werden kann.

In einer früheren Version wurde auch an die Einbettung von speziellen Konstrukten für die Beschreibung von HTML-Dokumenten gedacht. Dieser Gedanke wurde aber wieder verworfen, da aus Gründen der Einfachheit für die Dokumentanalyse bei der Suche in HTML-Dokumenten das Unix-Kommando ‚sgrep' [2] verwendet wird.

Derzeit wird die hier beschriebene Sprache für die Domäne deutscher Geschäftsbriefe eingesetzt, aus der auch die meisten Beispiele entnommen wurden.

2.0 Anforderungen

Im folgenden werden die Anforderungen aufgelistet, die beim Design der Sprache besonders berücksichtigt wurden.

- Die Sprache soll einfache syntaktische Strukturen auch einfach ausdrücken. Der Satz: „Die Katze beißt den Hund“ soll genauso einfach auch als Textpattern formuliert werden können. In solch einem Fall ist also keine Generalisierung des Satzes erwünscht.
- Verschachtelte Ausdrücke sollen formuliert werden können. Beispielsweise möchte man verschiedene Formen einer Anrede nur einmal formulieren (Herr, Frau, Dr,...), wobei diese aber wiederum in unterschiedlichen Sätzen auftreten können.
- Pattern als syntaktische Erscheinungsform von Informationen müssen mit ihrer Semantik verbunden werden können, um eine Extraktion von Informationen gewährleisten zu können. Die Auflösung von Widersprüchen, die aus widersprüchlichen extrahierten Informationen entstehen können, soll allerdings hier nicht berücksichtigt werden.
- Die Sprache soll die Verwendung von linguistischen Wissensquellen ermöglichen. Insbesondere ist an die Einbindung einer Morphologie und eines Thesaurus gedacht.
- Zu einem Pattern bzw. einer Gruppe von Pattern möchte man Klassen von Texten angeben, für die diese Pattern signifikant sind. Auch der Grad der Signifikanz soll formuliert werden können.
- Die Definition von beliebigen boole‘schen Patternkombinationen soll möglich sein.
- Die Beschreibungssprache soll den Erfordernissen der Dokumentanalyse Rechnung tragen. So sind z.B. Fontattribute oder auch Fehlertoleranz gegenüber OCR-Fehlern von Bedeutung.

3.0 Grundlegende Überlegungen

Um die verschiedenen Anforderungen erfüllen zu können, wird das in Abbildung 2 dargestellte hierarchische Szenario für die Einbindung der Patternbeschreibungssprache vorgeschlagen.

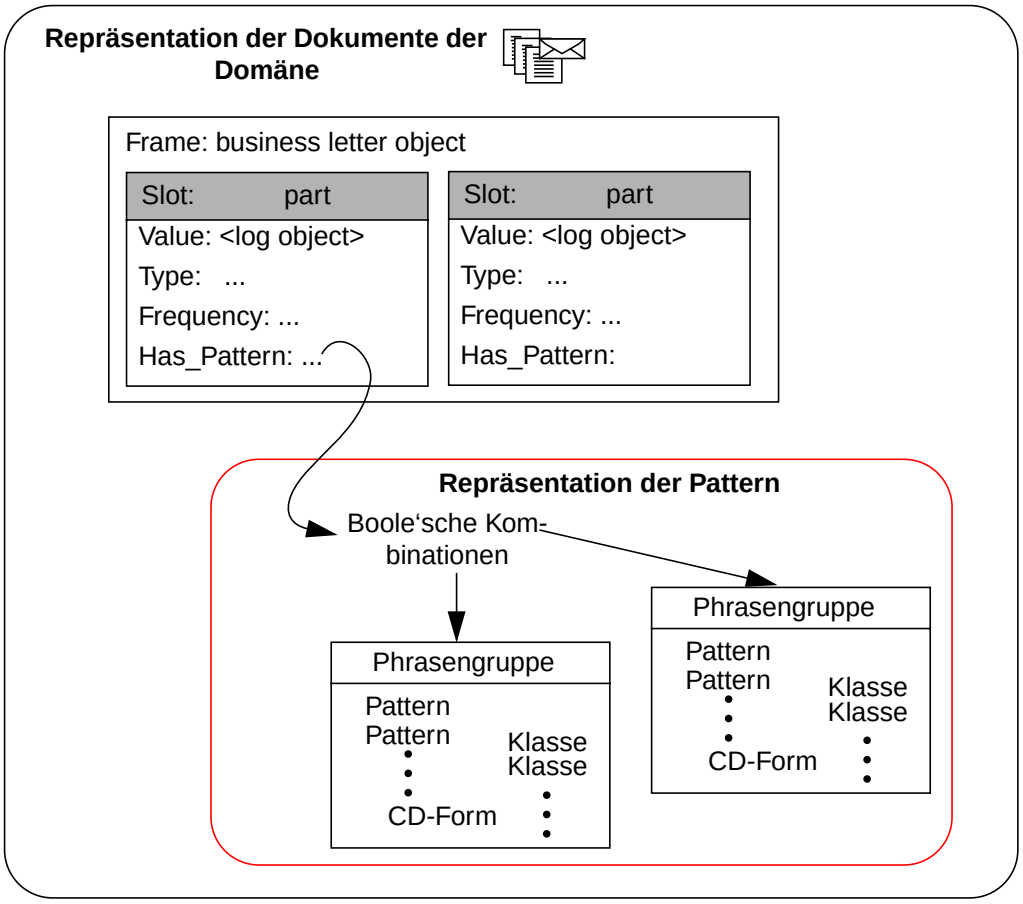

ABBILDUNG 2: Hierarchie der Patternsprache und deren mögliche Einbindung in eine übergeordnete Dokumentrepräsentationssprache

Es wird davon ausgegangen, daß einem Dokument eine Aggregationshierarchie zugeordnet wird. Innerhalb dieser Hierarchie werden die Pattern dann in dem Objekt eingebunden, in dem sie tatsächlich physikalisch auftreten. So kann man beispielsweise spezifizieren, daß das Pattern „Mit freundlichen Grüßen“ in einem Geschäftsbrief in der Grußformel auftritt. Im Beispiel der Abbildung können somit beliebige Schachtelungen von Pattern durch eine Facette mit Attributnamen „Has_Pattern“ in die globale Dokumentrepräsentation eingebunden werden.

Die Pattern selbst können (müssen aber nicht) innerhalb einer Phrasengruppe definiert werden. Solch eine Phrasengruppe soll Pattern mit gleicher Semantik zusammenfassen, so daß diesen dann eine gemeinsame Semantikbeschreibung zugewiesen werden kann. Zur Beschreibung der Semantik wählen wir *Conceptual Dependency Forms* – CDForms [3]. Zusätzlich kann zu einer Patterngruppe angegeben werden, inwieweit sie relevant für welche Klasse ist.

4.0 Stand der Kunst

Ähnliche Patternbeschreibungssprachen gibt es sowohl im Gebiet der NLP als auch in der Dokumentanalyse.

Innerhalb der NLP haben Hayes [4] und Gilardoni [5] Pattern-Matching zur Unterstützung der Klassifikation elektronischer Texte eingesetzt. Die dazu definierten Sprachen sind recht ähnlich. Das TCS System von Hayes enthält folgende Konstrukte auf Wortebene:

- Obligatorisches Wort
- Optionales Wort
- Stemming für Nomen und Verben
- Wildcards für Artikel, Zahlen und Interpunktionszeichen
- Negation

Wörter können verbunden werden durch Disjunktionen, Konjunktionen und Wortauslassungen (Skips). Desweiteren kann der Ort des Auftretens einer Wortsequenz spezifiziert werden, es kann angegeben werden, ob Groß-/Kleinschreibung relevant ist und Ausdrücke können hierarchisch verwendet werden. Für die Verbindung zur Klasse kann die Aussagekräftigkeit des Ausdrucks durch ein Gewicht spezifiziert sowie eine Mindesthäufigkeit des Auftretens des Ausdrucks im Text vorgegeben werden. Die Pattern werden innerhalb von Regeln abgearbeitet.

Innerhalb der Dokumentanalyse wurde bisher nur eine Patternbeschreibungssprache zur Beschreibung von Dokumenten durch Kerpedjev [6] vorgestellt. Sie enthält größtenteils eine Teilmenge der bereits vorgestellten Konstrukte (Wildcards für Ziffern und Interpunktion, Konjunktion, Disjunktion, Negation, Optionalität, Subpatterns, Wortwiederholungen, Angabe von Klassen, zu denen ein Wort gehören muß), wurde aber nicht speziell für die Dokumentanalyse angepaßt. Die Pattern werden zur Analysezeit in Transitionsgraphen für einen Parser übersetzt und durch einen Parser abgearbeitet.

Bayer und Walischewski [7] haben einen Sprache vorgestellt, die auf einem semantischen Netz beruht. Die Sprache gestattet jedoch nur den fehlertoleranten Wortvergleich mit einem Wörterbuch, wobei der Ort des Auftretens einer Wortfolge angegeben werden kann. Sie enthält also so gut wie keine Konstrukte zur Beschreibung von Wortsequenzen, berücksichtigt aber durch Fehlertoleranz eine Anforderung innerhalb der Dokumentanalyse. Bisher wurde die Sprache zur Extraktion von Absender, Empfänger, Datum und bestimmten Schlüsselwörtern aus Geschäftsbriefen eingesetzt.

Bisher existiert also noch keine Sprache, die die Belange der beiden Bereiche mehr als oberflächlich miteinander verknüpft.

5.0 Repräsentation der Syntax: Textpattern

Um Textteile syntaktisch beschreiben zu können, verwenden wir Textpattern, die ihrerseits wiederum aus Wortpattern aufgebaut sind.

- Ein *Wortpattern* beschreibt ein einzelnes Wort eines Textpatterns und kann Informationen darüber beinhalten, wann das Wortpattern erfüllt ist. Die Worteigenschaften werden in Form von null- oder einstelligen Schlüsselwörtern in Postfix-Notation spezifiziert.
- Ein *Textpattern* besteht im Wesentlichen aus einer Aneinanderreihung von Wortpattern und stellt eine syntaktische Beschreibung von Satzfragmenten dar.

5.1 Syntax von Wortpattern

Ein Wortpattern besteht aus zwei Teilen: Zunächst wird entweder das gesuchte Wort angegeben oder es wird eine Variable eingeführt für den Fall, daß die gefundene Instanz des Wortes extrahiert werden soll. Eine Variable wird durch ein führendes Fragezeichen gekennzeichnet (Syntax: `?<string>`) und hat den Vorteil, daß sie in der semantischen Beschreibung des Textpatterns verwendet werden kann.

Der zweite Teil des Wortpatterns besteht aus einer geklammerten Liste aus Schlüsselwörtern und deren Werten. Schlüsselwörter können null- oder einstellig sein. Durch sie werden Eigenschaften definiert, die für das Wort selbst gelten sollen. Die Verwendung der Klammern zur Kapselung der Schlüsselwortliste dient der besseren Lesbarkeit des Wortpatterns. Da die Klammern für die Abarbeitung nicht notwendig sind, ist ihre Verwendung optional.

Daraus ergibt sich folgende Form eines Wortpatterns:

```
Form: <string> (<Schlüsselwort1> <pot. Wert des Schlüsselworts>
                <Schlüsselwort2> …)
```

Die Reihenfolge der Schlüsselwörter ist irrelevant. Besteht ein Wortpattern lediglich aus der Angabe eines Strings, werden für die Schlüsselwörter die Defaultwerte angenommen. In diesem Fall findet ein exakter Vergleich von Wortvollformen statt.

5.2 Schlüsselwörter für Wortpattern

Schlüsselwörter für Wortpattern lassen sich zum Teil bezüglich unterschiedlicher Eigenschaften einteilen. *Wortgeneralisierungen* erweitern den Suchraum für den String selbst, *Wortspezialisierungen* schränken ihn ein. Die meisten der *sonstigen Schlüsselwörter* beeinflussen den Suchraum für das Textpattern an der Stelle, an der das Wort auftritt.

5.2.1 Wortgeneralisierungen

- **:substring** spezifiziert ein Teilstringmatching. Dies bedeutet, daß das zugehörige Wort (= `<string>` in obiger Beschreibung) nur ein Teilwort eines im Text zu suchenden Wortes darstellt.
- **:tolerance <Levenshtein-Distanz>** gibt die Toleranz des Matchings an. Die absolute *Levenshtein-Distanz* [9] drückt die Ähnlichkeit zweier Strings aus. Ihr Definitionsbereich liegt im Intervall [0;∞] und gibt an, wieviele Buchstabenersetzungen, Einfügungen und Auslassungen pro Wort vorkommen dürfen. Der Defaultwert für das Matching ist null, so daß dann ein exakter Vergleich stattfindet.
- **:morph_stem** gibt an, daß zum zugehörigen Wort die Stammform berechnet werden soll oder das Wort eine Stammform darstellt und dann für alle Wörter ein Wortvergleich auf

Stammformebene stattfindet. Im Text darf also jede beliebige Vollform dieser Stammform stehen.

- **:morph_compound_part** gibt an, daß das zugehörige Wort ein Teilwort eines Kompositums ist. Im Text dürfen also alle Komposita stehen, die das Teilwort beinhalten.
- **:morph_cat <cat>** spezifiziert die morphologische Kategorie, zu der das Wort gehören soll. Dieses Schlüsselwort darf nur als Schlüsselwort für eine Variable verwendet werden. Das tatsächlich vorkommende Wort wird extrahiert. Die potentiellen Kategorien werden durch die eingesetzte Morphologie definiert. Die für die Geschäftsbriefdomäne verwendete Morphologie Morphic-Plus [8] hat 22 Kategorien beispielsweise für Nomen, Verben, Adjektive, Kardinalzahlen, Stopwörter, Namen, Abkürzungen …
- **:hyperonym <cat>** spezifiziert die semantische Kategorie, zu der das Wort gehören soll. Dieses Schlüsselwort darf nur als Schlüsselwort für eine Variable verwendet werden. Das tatsächlich vorkommende Wort wird extrahiert. Semantische Kategorien können in einem Lexikon bzw. Thesaurus definiert (z.B. Produkte) oder durch einen Automaten spezifiziert sein (z.B. Datum).
- **:synonyms** spezifiziert, daß an dieser Stelle auch alle Synonyme des Wortes vorkommen dürfen. Wenn das Wort in mehreren Synonymgruppen eingetragen ist, werden alle Gruppen zum Vergleich herangezogen.
- **:phrase_group** spezifiziert, daß an dieser Stelle alle Vorkommen von Pattern aus einer bestimmten Phrasengruppe stehen dürfen. Das zugehörige Wort muß als Name einer Phrasengruppe bekannt sein. Mit diesem Schlüsselwort können also hierarchische Pattern formuliert werden können.

5.2.2 Wortspezialisierungen

- **:structure <reg_exp>** darf nur bei Angabe einer Variable verwendet werden. Mit diesem Feature kann ein regulärer Ausdruck für die Syntax der Variable definiert werden (sinnvoll z.B. bei Datumsangaben).
- **:font_size <size>** gibt die relative Größe des für das Wort verwendeten Fonts an. Wird dieses Feature nicht spezifiziert, ist die Größe des Fonts für den Match irrelevant. Mögliche Werte dieses Aufzählungstyps sind small, regular und large. Die Festlegung, wann ein Font z.B. small ist, muß an anderer Stelle erfolgen. Sie kann beispielsweise relativ zum aktuellen Dokument berechnet oder absolut durch ein Intervall der Punktgröße spezifiziert werden.
- **:font_spacing <size>** gibt die relative Größe des Freiraums zwischen den einzelnen Zeichen an. Wird dieses Feature nicht spezifiziert, ist der Abstand der Zeichen für den Match irrelevant. Mögliche Werte dieses Aufzählungstyps sind small, regular und large.
- **:font_weight <font_name>** gibt die relative Stärke des Fonts an. Wird dieses Feature nicht spezifiziert, ist die Stärke für den Match irrelevant. Mögliche Werte dieses Aufzählungstyps sind regular, bold und italic.
- **:underlined** gibt an, daß das Wort im Text unterstrichen vorkommen soll und kann z.B. bei Überschriften sinnvoll sein.

5.2.3 Sonstige Schlüsselwörter für Wortpattern

- **:distance_to_succ_horizontal <distance>** gibt den maximalen Wortabstand des Wortpatterns zum nachfolgenden Wortpattern in horizontaler Richtung an. Der Defaultwert 0 bestimmt, daß Wortpattern im Text direkt aufeinander folgen müssen.

- **:distance_to_succ_vertical <distance>** gibt den maximalen Wortabstand des Wortpatterns zum nachfolgenden Wortpattern in vertikaler Richtung an. Der Defaultwert 0 bestimmt, daß Wortpattern im Text direkt aufeinander folgen müssen.
- **:optional** gibt an, daß das Wortpattern optional ist, d.h. es kann, aber muß nicht vorkommen. Kommt es nicht vor, beziehen sich Abstandsangaben im Vorgängerwortpattern auf das Nachfolgerwortpattern.
- **:search_direction <direction>** gibt die Suchrichtung für das nachfolgende Wort an. Mögliche Werte sind horizontal, vertical, all. Hiermit kann eine globale Einstellung überdefiniert werden. Je nach Wert stellt dieses Schlüsselwort entweder eine Spezialisierung oder Generalisierung dar.
- **:not** gibt an, daß das Wort an der Stelle nicht stehen darf.
- **:repeat <n>** gibt an, daß das Wort an dieser Stelle mindestens einmal und bis zu n-mal stehen darf. In Verbindung mit dem Schlüsselwort :optional kann das Wort dann auch überhaupt nicht vorkommen. Alle anderen Schlüsselwörter müssen für jedes Vorkommen erfüllt sein.
- **:case_sensitivity <true|false>** gibt an, ob Groß-/Kleinschreibung für den Wortmatch relevant ist. Hiermit wird für das Wort eine mögliche globale Einstellung überdefiniert. Siehe dazu auch Kapitel 8.0.
- **:strip <true|false>** gibt an, ob in dem zu findenden Wort für den Wortvergleich Interpunktionszeichen u.ä. entfernt werden sollen. Hiermit wird für das Wort eine mögliche globale Einstellung überdefiniert. Siehe dazu auch Kapitel 8.0.

5.3 Verknüpfung von Schlüsselwörtern

Werden mehrere Schlüsselwörter für ein Wort spezifiziert, so sind diese Schlüsselwörter gewöhnlich konjunktiv miteinander verknüpft, d.h. alle Bedingungen müssen erfüllt sein. In manchen Fällen ist die genaue Bedeutung einer Schlüsselwortverknüpfung allerdings unklar bzw. die Verknüpfung ist nicht sinnvoll und wird daher ausgeschlossen.

Es folgt ein Auflistung der Semantik in unklaren Fällen:

- Werden :synonyms und :morph_stem verwendet, bedeutet dies, daß alle Vollformen aller Synonyme der Stammform des Wortes im Pattern im Text auftreten dürfen.
- Werden :hyperonym und :morph_stem verwendet, bedeutet dies, daß alle Vollformen aller Begriffe, die unter dem Oberbegriff zusammengefaßt sind, s im Pattern im Text auftreten dürfen.
- Werden :repeat und :search_direction oder :repeat und :distance_to_succ_... verwendet, gelten Richtungs- und Abstandsangabe sowohl für die Mehrfachvorkommen des Wortpatterns als auch für die Beziehung zum nachfolgenden Wortpattern.

Nun folgt eine Auflistung von nicht erlaubten Kombinationen von Schlüsselwörtern:

- :substring, :morph_stem und :morph_compound_part schließen sich gegenseitig aus.
- :tolerance kann zwar mit :morph_stem und :morph_compound_part benutzt werden, aber es hängt natürlich von den Fähigkeiten der eingesetzten Morphologie ab, ob diese Bedingung tatsächlich abgeprüft werden kann.
- :synonyms schließt :substring und :morph_compound aus.
- :phrase_group: Mögliche Schlüsselwörter sind hier :search_direction, :distance_to_succ_horizontal, :distance_to_succ_vertical. Alle anderen Schlüsselwörter dürfen nicht gemeinsam mit :phrase_group verwendet werden.

- :morph_cat, :hyperonym: Da hier Informationen extrahiert werden, dürfen beide Schlüsselwörter nur als Schlüsselwörter einer Variable verwendet werden. Die Toleranz des Matchings wird durch die Möglichkeiten des überprüfenden Verfahrens bestimmt. Erlaubt sind alle Kombinationen außer mit :phrase_group.
- :not darf nicht mit :repeat verwendet werden.
- Wenn ein Wortpattern eine Variable und die :repeat Eigenschaft enthält, bedeutet dies, daß mehrere Informationen der gleichen Art (also z.B. Preise) an dieser Stelle extrahiert werden können. In diesem Fall werden alle Informationseinheiten, durch Leerzeichen getrennt, in dieser einen Variable gespeichert.

5.4 Syntax für Schlüsselwörter auf Textpatternebene

Wortpattern werden zu (Text-)Pattern zusammengefaßt. Dabei spiegelt die Reihenfolge der Wortpattern das Auftreten im Text wider.

```
Form: "<Wortpattern1> <Wortpattern2> …"
```

Für diese syntaktische Beschreibung können nun wiederum Schlüsselwörter spezifiziert werden.

```
Form: (:pattern "<Wortpattern1> <Wortpattern2> …"
                (<Schlüsselwort1> <Wert des Schlüsselworts> …
                 <Schlüsselwortn> <Wert des Schlüsselworts>)
      )
```

Das Schlüsselwort :pattern sowie die äußere Klammerung sind optional und dienen der besseren Lesbarkeit.

Die im Folgenden aufgelisteten Schlüsselwörter dürfen - falls nicht anders angegeben - höchstens einmal pro Textpattern stehen. Ihre Reihenfolge ist nicht von Bedeutung.

- **:search_space_left_boundary <number>**
- **:search_space_right_boundary <number>**
- **:search_space_upper_boundary <number>**
- **:search_space_lower_boundary <number>**

Diese vier Schlüsselwörter geben die maximalen räumlichen Ausdehnungen des Patterns nach links, rechts, oben und unten an und beschränken somit die absolute Lage des Patterns im Dokument. Dabei erfolgt die Zahlenangabe in Weltkoordinaten und Pixeln, wobei der Ursprung (0,0) sich in der linken oberen Ecke befindet.

- **:max_vertical_extension <number>**: Dieses Schlüsselwort gibt an, über wieviele Zeilen das Pattern sich maximal erstrecken darf. Es ist nur dann sinnvoll, wenn eine vertikale Suche erlaubt ist. Es darf nur einmal pro Pattern verwendet werden.
- **:logical_object <logical_object_1> …<logical_object_n>**: Dieses Schlüsselwort gibt die logische Lage eines Patterns im Dokument an. Erlaubte Logikobjekte sollten entweder in der übergreifenden Dokumentrepräsentationssprache spezifiziert sein oder in einer Ressource-Datei angegeben werden. Wird das Schlüsselwort mit mehreren Werten belegt, so entspricht dies einer Disjunktion der möglichen Logikobjekte.
- **:page <page_number>** schränkt die Suche nach dem Vorkommen des Patterns auf eine bestimmte Seite ein.

5.5 Festgelegte Abkürzungen und Defaultwerte

Für längere Schlüsselwörter ist in manchen Fällen eine schnelle, effiziente Schreibweise wünschenswert. Diese Abkürzungen können dann bei Bedarf zu ihrer Langform expandiert werden, um z.B. eine bessere Lesbarkeit zu gewährleisten. Die folgende Tabelle enthält eine Auflistung erlaubter Abkürzungen sowie die Defaultwerte für die Schlüsselwörter.

Schlüsselwort Wortpattern	Abkürzung	Defaultwert
:substring	:sub	– (= Eigenschaft egal)
:tolerance	:tol	0
:morph_stem	:ms	–
:morph_compound_part	:mcp	–
:morph_cat	:mc	–
:hyperonym	:hyp	–
:synonyms	:syn	–
:phrase_group	:pg	–
:structure	:struct	–
:font_size	:size	–
:font_spacing	:spacing	–
:font_weight	:weight	–
:underlined	:ul	–
:distance_to_succ_horizontal	:disth	0
:distance_to_succ_vertical	:distv	0
:optional	:opt	–
:search_direction	:search	global definiert
:not	:not	–
:repeat	:repeat	1
:case_sensitivity	:case	global definiert
:strip	:strip	global definiert

Schlüsselwort Textpattern	Abkürzung	Defaultwert
:search_space_left_boundary	:sleft	0
:search_space_right_boundary	:sright	Rechtes Dokumentende
:search_space_upper_boundary	:sup	0
:search_space_lower_boundary	:slow	Unteres Dokumentende
:max_vertical_extension	:max_ext	Seitenende
:logical_object	:log	–
:page	:page	–

5.6 Beispiele

Bsp1: Drei Phrasen sollen in einem Pattern formuliert werden:

Phrase1: "Bitte senden sie mir ..."

Phrase2: "Bitte schick mir heute die neuesten Kataloge..."

Phrase3: "Schick mir den neuesten Katalog ..."

-> Pattern in Langform:

(:pattern "Bitte (:optional) schicken (:synonyms :morph_stem :distance_to_succ_horizontal 1) mir (:distance_to_succ_horizontal 1) neuesten katalog (:morph_stem)")

-> Pattern in Kurzform:

"Bitte :opt schicken :syn :disth :ms1 mir :disth1 neuesten katalog :ms"

Bsp2: Pattern kennzeichnet eine Spalte mit bis zu 20 Preisangaben:

-> Pattern in Langform:

(:pattern "?preis (:repeat 20 :hyperonym Preis :search_direction vertical)")

-> Pattern in Kurzform:

"?preis :repeat 20 :hyp Preis :search vertical"

In der Variable ?preis sollte nach Abarbeitung des Patterns eine Liste mit Preisen stehen.

Bsp3: Dokumentdatum soll rechts oben gefunden werden

-> Pattern in Langform:

```
(:pattern "?ort  (:hyperonym Städtenamen :strip true) den ?datum (:hyperonym Datum)"
                (:search_space_left_boundary 1200
                 :search_space_right_boundary 2300
                 :search_space_upper_boundary 300
                 :search_space_lower_boundary 1200
                 :page 1)
```

-> Pattern in Kurzform:

"?ort :hyp Städtenamen :strip true den ?datum :hyp Datum" (:sleft 1200 :sright 2300 :sup 300 :slow 1200 :page 1)

Das explizite "strip" im Städtenamen gewährleistet die Entfernung eines Kommas hinter der Ortsangabe.

6.0 Repräsentation der Semantik: CD-Formen

Die Semantik von Textpattern wird durch eine CD-ähnliche Struktur repräsentiert, wobei Spezialisierungen für die Domäne der Geschäftsbriefe vorgenommen wurden. Conceptual dependency (CD) ist eine Theorie der maschinellen Verarbeitung natürlicher Sprache, die zum ersten Mal 1972 von Schank [3] eingeführt wurde. Ihr Anspruch liegt darin, natürlichsprachlichen Äußerungen eine Repräsentationsform zuzuweisen, die maschinell verarbeitet werden kann.

Wir wählen diesen Formalismus, da er für die Beschreibung von Textpattern ausdrucksmächtig genug ist. An dieser Stelle soll nicht näher auf die Theorie eingegangen werden, da sich eine sehr gute Beschreibung in [10] findet.

Prinzipiell sind zwei Arten von Strukturen möglich, je nachdem, ob ein Satz einen Sachverhalt oder eine Aktion beschreibt.

- Statische CD-Formen: Sie beinhalten Hypothesen über Eigenschaften oder Zustände von Objekten. Deshalb enthalten sie als zentrales Element das Objekt.
- Aktive CD-Formen: Durch sie werden allgemeine Hypothesen über die Elemente von sprachlichen Äußerungen repräsentiert. Für die Geschäftsbriefdomäne schlagen sich solche sprachlichen Äußerungen im Briefinhalt nieder. Aktive CD-Formen beschreiben eine Veränderung des Kontextes durch Aktionen. Deshalb wird ihnen als zentrales Element immer eine Aktion zugeordnet.

CD-Strukturen bestehen aus einer geschachtelten, hierarchischen Anordnung von Attribut-Wert-Paaren. Dabei werden Attribute immer kleingeschrieben und Werte immer in Anführungszeichen gesetzt.

Auf der obersten Ebene wird spezifiziert, ob es sich um eine Aktion oder ein Objekt handelt. Im ersten Fall handelt es sich dann um eine aktive CD-Form, im zweiten Fall um eine statische. Dann folgt eine Schachtelung weiterer Attribut-Wert Paare. Obligatorisch ist die Angabe des Typs der Aktion bzw. des Objekts. Alle anderen Attribute sind optional und dürfen mehrfach vorkommen. Sie enthalten entweder einen Wert oder eine geschachtelte Struktur von Attribut-Wert Paaren.

Um CDs für die Informationsextraktion nutzen zu können, erlauben wir die Verwendung von Variablen an Stelle von elementaren Werten. Variablen werden durch ein führendes Fragezeichen gekennzeichnet und werden nicht in Anführungszeichen gesetzt. Beispiele für die Verwendung von Variablen folgen in Kapitel 7.3. Dadurch, daß in der Definition von CD-Formen die gleichen Variablen wie in Textpattern verwendet werden können, ist eine Übertragung der extrahierten Information an die passende Stelle der Semantik möglich.

6.1 Statische CD-Formen

Ein Objekt wird durch das obligatorische Attribut **typ** beschrieben, das die Art des Objektes angibt. Alle anderen Attribute sind optional. Da die Eigenschaften eines Objektes domänenabhängig sind, geben wir hier nur eine Aufzählung von Beispielattributen.

Möchte man beispielsweise einen eingehenden Geschäftsbrief näher beschreiben, so kann man sich folgende CD-Form vorstellen:

```
(objekt ((typ <Nachrichtentyp>)
         (nummer <string>)
         (medium <Mediumangabe>)))
```

mit

<Nachrichtentyp> ::= "Anfrage"|"Angebot"|"Auftrag"|

Auftragsbestätigung"|"Bedarfsanmeldung"|"Gutschrift"|
"Lieferschein"|"Mahnung"|"Rechnung"|"Werbung"
<Mediumangabe> ::= "Fax" | "Papier" | "Telefon"

Möchte man beispielsweise einen bestimmten Rabatt in einem Geschäftsbrief näher beschreiben, so kann man sich folgende CD-Form vorstellen:

(objekt ((typ <Rabattyp>)
(rabatt <integer>)
(maß <Maßangabe>)
(mindestmenge <integer>)
(einheit <Einheitsangabe>

mit
<Maßangabe> ::= "DM" | "Prozent"
<Rabattyp> ::= "Skonto"|"Hochschulrabatt"|"Sonderrabatt"
<Einheitsangabe> ::= "Stück"| ...

6.2 Aktive CD-Formen

Folgende Attribute sind erlaubt, um eine aktive CD-Form zu beschreiben:

- **typ:** gibt das Verb an, das die zugrundeliegende Handlung charakterisiert und ist obligatorisch.
- **zeitbezug:** enthält entweder ein Datum oder die Angabe eines Zeitraumes.
- **ortsbezug:** enthält entweder eine Orts- oder eine Richtungsangabe.
- **agent:** spezifiziert eine beteiligte Person (auch im juristischen Sinn, z.B. Firma).
- **objekt:** gibt im weitgefaßten Sinne das Objekt an.

Im folgenden werden die erlaubten Werte für die Attribute näher beschrieben:

- typ ::= "beziehen" | "schreiben" | "zahlen" | ...
  Diese Liste wird nicht vollständig angegeben, da sie domänenabhängig ist. Möchte man eine Domänenunabhängigkeit gewährleisten, kann man die erlaubten Aktionen auch auf die in [10] angegebenen Aktionen einschränken.
- zeitbezug

  (zeitbezug ((tag <tagangabe>)
  (monat <monatsangabe>)
  (jahr <jahresangabe>)
  (typ <typangabe>)))

  mit
  <tagangabe> ::= <zweistell. Integer> | "akt_Tag"
  <monatsangabe> ::= <zweistell. Integer> | "akt_Monat"
  <jahresangabe> ::= <vierstell. Integer> | "akt_Jahr"
  <typangabe> ::= "Datum"|"Zeitraum"
- ortsbezug

  (ortsbezug <ortsangabe>)

  mit <ortsangabe>::= "Kaiserslautern"|....
- agent

(agent ((name <name>)
(partei <parteiangabe>)
(telefon <telefonnummer>)
(rolle <rollenangabe>))

mit

<rollenangabe> ::= "Sekretär"|"Kaufmann"|"Techniker"

<parteiangabe> ::= "Sender"|"Empfänger"

- objekt: Die Belegung des Objekt-Attributs erfolgt wie in Kapitel 6.1 beschrieben.

Die folgenden Beispiele verdeutlichen nun das Verständnis für CD-Formen.

6.3 Beispiele

Die beiden ersten Beispiele wurden der Domäne der Geschäftsbriefe entnommen.

Bsp1: Text "Ihr Angebot Nr. 0815 vom 9.9.99"

```
:cd-form (aktion ((typ "beziehen")
                  (objekt  ((typ "Angebot")
                            (nummer "0815")
                            (zeitbezug ((tag 09)
                                        (monat 09)
                                        (jahr 1999)
                                        (typ "Datum")))))
          )
```

Der Agent des Objektes (= Empfänger) kann ohne Kontextwissen aus dem Pronomen "Ihr" nicht ermittelt werden.

Bsp2: Text "Lieferschein Nr.007" ;;dazu gibt es zwei Hypothesen, eine aktive und eine statische

```
:cd-form (aktion ((typ "beziehen")
                  (objekt ((typ "Lieferschein")
                           (nummer "007")))))
         (objekt ((typ "Lieferschein")
                  (nummer "007")))
```

Bsp3: Text "Die Katze Pussi beißt den alten Hund"

```
:cd-form (aktion ((typ "beißen")
                  (agent ((typ "Katze")
                          (name "Pussi")))
                  (objekt ((typ "Hund")
                           (eigenschaft "alt")))))
```

7.0 Repräsentation von Syntax und Semantik: Die Phrasengruppe

7.1 Schlüsselwörter für eine Phrasengruppe

Eine Phrasengruppe besteht aus einer Menge von Textpattern, die alle die gleiche Semantik besitzen.

```
Form: (:name_of_phrasegroup <name>
           <Textpattern1> … <Textpattern n>
           (<Schlüsselwort1> <Wert des Schlüsselworts> …
            <Schlüsselwortn> <Wert des Schlüsselworts>)
      )
```

Dabei sind das Schlüsselwort :name_of_phrasegroup, der Name der Phrasengruppe und die äußere Klammerung optional.

Die Reihenfolge der Schlüsselwörter ist irrelevant.

- **:cd_form <cd_form>** gibt die Semantik des Textpatterns an (siehe Kapitel 6.0). Pro Phrasengruppe darf höchstens eine CD-Form angegeben werden.
- **:category <name> <value>** dient der Angabe einer Kategorie mit Wahrscheinlichkeitsmaß. Das Wahrscheinlichkeitsmaß kann entfallen. Dieses Schlüsselwort ist optional, kann aber auch mehrfach vorkommen.
  Zusätzlich darf das Schlüsselwort :category auch auf höheren Schachtelungsebenen vorkommen (z.B. für eine Konjunktion von Phrasengruppen).
  Die Angabe der Kategorie kann in zweierlei Hinsicht genutzt werden:
  - als einfache Dokumentklassifikation
  - Wenn die Kategorie bereits feststeht, können gezielt nur die Pattern gesucht werden, die zu dieser Kategorie gehören.

Die explizite Darstellung der Schachtelungsebenen Phrasengruppe und Textpattern wurde optional definiert, um auch eine einfache Formulierung von boole‘schen Kombinationen von Phrasengruppen und Pattern zu ermöglichen.

Zu beachten ist, daß bei Auslassung der Schlüsselwörter :phrase_group und :pattern und der dazugehörigen Klammern eine strukturelle Auftrennung der übrigen Schlüsselwörter in Schlüsselwörter für Phrasengruppen und Textpattern nicht mehr möglich ist. Die Schlüsselwörter sind in diesem Fall also nur durch ihre konkrete Syntax unterscheidbar.

7.2 Operatoren über Phrasengruppen und Textpattern

Wenn man Patternausdrücke beliebig schachteln möchte, um beispielsweise eine Textklassifikation mit einem komplexen Ausdruck durchzuführen, benötigt man boole’sche Operationen über den bisher eingeführten Konstrukten.

- **and**: Dieser Operator verlangt das Auftreten aller Pattern oder Phrasengruppen.
- **or**: Dieser Operator verlangt das Auftreten mindestens eines Patterns oder einer Phrasengruppe.
- **not:** Dieser Operator entspricht NICHT dem in Kapitel 5.2.3 angegebenen Schlüsselwort für Wortpattern :not. Er verlangt, daß keine der folgenden Textpattern oder Phrasengruppen im Text auftreten dürfen.

Diese Operatoren dürfen in beliebigen Schachtelungen von Textpattern und Phrasengruppen auftreten, allerdings darf ein Operator sich immer nur entweder auf Pattern oder auf Phrasengruppen beschränken. In Anlehnung an die Syntax von Textpattern werden die Operatoren in Infix-Notation verwendet. Die Art der Schachtelung muß explizit durch Klammern vorgegeben werden.

7.3 Beispiele

Bsp1: Beispieltext "Würden Sie mir 5 Schreibmaschinen schicken?"

-> Langform der Beschreibung:

```
(:name_of_phrasegroup Anfrageformulierung
        (:pattern
            "Würden Sie mir ?zahl (:morph_cat kardinalzahl)
             ?bestellartikel (:hyperonym bestellartikel :morph_stem) schicken (:strip true)"
            (:logical_object body))
        (:pattern
            "Bitte send (:morph_stem :distance_to_successor_horizontal 1) mir ?zahl
             (:morph_cat kardinalzahl) ?bestellartikel (:hyperonym bestellartikel
              :morph_stem) zu")
        (:cd-form
            (aktion ((typ "anfragen")
                        (objekt ((aktion "zusenden")
                                    (objekt ?bestellartikel)
                                    (menge ?zahl))
                        )
                    )
            ))
        (:category Anfrage 0.5)
)
```

-> Kurzform der Beschreibung:

```
"Würden Sie mir ?zahl (:mcat kardinalzahl) ?bestellartikel (:hyp bestellartikel :ms) schicken (:strip
    true)" (:log body)
"Bitte send (:ms :disth 1) mir ?zahl (:mcat kardinalzahl) ?bestellartikel (:hyp bestellartikel :ms) zu")
(:cd-form (aktion ((typ "anfragen")
                    (objekt ((aktion "zusenden")
                                (objekt ?bestellartikel)
                                (menge ?zahl))
                    )
          ))
 (:category Anfrage 0.5)
```

Bsp2: Spalte mit bis zu 20 Preisangaben soll gefunden werden und die Preise sollen extrahiert werden

-> Langform der Beschreibung:

```
(:name_of_phrasegroup Preisspalte
        (:pattern "?preis (:repeat 20 :hyperonym Preis :search_direction vertical)")
         (:cd-form (objekt ((typ "Preisliste")
                            (werte ?preis)))
        )
)
```

Bsp3: In einem Text sollen entweder Vorkommen von "Bank" und "Geld" gefunden werden, aber der Text darf keine Vorkommen von "Spaziergängern" enthalten oder der Text soll Vorkommen von "Sparkassen" enthalten. Aus den Vorkommen soll auf die Kategorie "Geldbank", also eine Bank, die mit Geld zu tun hat, geschlossen werden.

-> Kurzform der Beschreibung:

```
(or (and "Bank (:ms)"  "Geld (:ms)"  (not "Spaziergänger (:ms)")
     "Sparkasse (:ms)"
)
(:category Geldbank 0.5)
```

8.0 Globale Definitionen

Damit die vorgeschlagene Beschreibungssprache auch abgearbeitet werden kann, ist es notwendig, bestimmte Parameter global zu definieren. Wir schlagen folgende Parameter vor:

- **lex_of_synonyms <Pfad>** gibt den Pfad für das zugrundeliegende Synonym-Wörterbuch an.
- **lex_of_hypernyms <Pfad>** gibt den Pfad für das Wörterbuch für semantische Kategorien an.
- **strip_set <Interpunktionszeichen>** gibt eine Menge von Zeichen an, die bei der Verarbeitung eines Textes am Wortanfang und -ende abgeschnitten werden können.
- **strip <true|false>** gibt an, ob die durch strip_set definierte Menge im aktuell bearbeiteten Dokument abgeschnitten werden soll.
- **case_sensitivity <true|false>** gibt an, ob beim Matching Groß/Kleinschreibung relevant sind oder nicht. Ist no spezifiziert, so werden Pattern und Text in Kleinbuchstaben transformiert.
- **horizontal_search <true|false>** gibt an, ob eine horizontale Suchrichtung inkl. Zeilenumbruch am Zeilenende erwünscht ist.
- **vertical_search <true|false>** gibt an, ob eine vertikale Suchrichtung erwünscht ist.
- **read_from_ascii <true|false>**: Dokumente sind nur als Ascii-Text vorhanden, somit werden viele Worteigenschaften uninteressant, z.B. eine vertikale Suche.
- Definition der Aufzählungstypen für die Fontattribute:, z.B. wie lautet der Wertebereich für einen kleinen Font in Anzahl der Pixel, was bedeutet small_spacing...
- **convert_umlaut <true|false>** gibt an, ob Umlaute und „scharfes s“ in Text und Pattern konvertiert (transskribiert) werden sollen.

9.0 Sonderzeichen

- **()** Ein rundes Klammerpaar umschließt die Angabe von Attributen (= Schlüsselwort oder Paar aus Schlüsselwort und Wert).
- **:** Ein Doppelpunkt kennzeichnet ein Schlüsselwort und steht als erstes Zeichen innerhalb des Schlüsselwortes.
- **"** Anführungszeichen dienen als Begrenzer eines Textpatterns und kennzeichnen Anfang und Ende.
- **?** Ein Fragezeichen kennzeichnet eine Variable und steht als erstes Zeichen innerhalb eines Variablennamens.
- **;;** Zwei Semikoli werden verwendet für Kommentare, die dann bis zum Zeilenende reichen dürfen.
- **** Ein Backslash steht innerhalb von Wortpattern und innerhalb des Namens einer Kategorie vor Sonderzeichen im String.
- Auf ein Konstrukt zum Schützen eines ganzen Wortes wurde verzichtet.

Anhang

BNF-Notation der Syntax (ohne Abkürzungen)

<classified_expression> ::= <expression> :category <category_identifier> {<category_probability>}

<expression> ::= <phrasegroup>
| not <expression>
| and <expression> <expression> {<expression>}
| or <expression> <expression> {<expression>}
| "(" <expression> ")"

<phrasegroup> ::= "(" :name_of_phrasegroup <phrase_identifier> <textpattern> {<textpattern>}
{ "(" <phrasefeature> ")" } ")"
| <phrase_identifier> <textpattern> {<textpattern>} { "(" <phrasefeature> ")" }

<phrasefeature> ::= :cd_form <cd_form>
| :category <category_identifier> {<category_probability>}

<textpattern> ::= "(" :pattern "\"" <wordpattern> {<wordpattern>} "|"" "(" {<textfeature>} ")" ")"
| (" :pattern "\"" <wordpattern> {<wordpattern>} "\"" {<textfeature>} ")"
| "\"" <wordpattern> {<wordpattern>} "\"" "(" {<textfeature>} ")"
| "\"" <wordpattern> {<wordpattern>} "\"" {<textfeature>}

<textfeature> ::= :search_space_left_boundary <integer>
| :search_space_right_boundary <integer>
| :search_space_upper_boundary <integer>
| :search_space_lower_boundary <integer>
| :max_vertical_extension <integer>
| :logical_object <logical_object_identifier>
| :page <integer>

<wordpattern> ::= <word> "(" {<wordfeature>} ")"
| <word> {<wordfeature>}

<wordfeature> ::= :substring
| :tolerance <integer>
| :morph_stem
| :morph_compound_part
| :morph_cat <morph_cat_identifier>
| :hyperonym <sem_cat_identifier>
| :synonyms
| :phrase_group
| :strip <boolean>

| :structure <reg_exp>
| :case_sensitivity <boolean>
| :font_size <size_identifier>
| :font_spacing <size_identifier>
| :font_weight <weight_identifier>
| :underlined
| :distance_to_succ_horizontal <integer>
| :distance_to_succ_vertical <integer>
| :optional
| :search_direction <direction_identifier>
| :not
| :repeat <integer>

<category_identifier>::= <string> ;;Vordefinierte Klassen, für Geschäftsbriefe bspw.
Anfrage | Bestellung | Auftragsbestätigung | Mahnung | Rechnung | ...

<category_probability>::= {0;1]

<phrase_identifier> ::= <string> ;; Bedeutungstragender, eindeutiger Name für eine Phrasengruppe

<morph_cat_identifier> ::= <string> ;; Morphologische Kategorie z.B. Verb, Nomen, ...

<sem_cat_identifier> ::= <string> ;; Oberbegriff für eine Gruppe von Wörtern z.B. Datum,
Städtenamen,...

<reg_exp> ::= Die Syntax eines regulären Ausdrucks wird hier nicht näher spezifiziert, da sie wahrscheinlich durch den Einsatz vorhandener Software bestimmt wird z.B. durch das Unix-Kommando ’grep’.

<size_identifier> ::= small | regular | large

<weight_identifier> ::= regular | italic | bold

Literatur

[1] Claudia Wenzel. Supporting Information Extraction from Printed Documents by Lexico-Semantic Pattern Matching. *Proc. of the Fourth International Conference on Document Analysis and Recognition (ICDAR 97)*, Ulm, Deutschland, August 1997.

[2] Sgrep home page: http://www.cs.helsinki.fi/~jjaakkol/sgrep.html

[3] Roger C. Schank. Conceptual Dependency: A theory of natural language understanding. *Cognitive Psychology*, 3 (4), Seiten 552-631, 1972, zitiert in Encyclopedia of Artificial Intelligence.

[4] P. J. Hayes, P. M. Andersen, I. B. Nirenburg, L. M. Schmandt. TCS: A Shell for Content-Based Text Categorization. *Proc. of the Sixth Conference on AI Applications*, Santa Barbara, CA, USA, 1990, Seiten 320-326.

[5] L. Gilardoni, P. Prunotto, G. Rocca. Hierarchical Pattern Matching for Knowledge Based News Categorization. *Proc. of the RIAO 94*, Rockefeller University, New York, USA, Oktober 1994, Band 1, Seiten 67-81.

[6] S. M. Kerpedjiev. Automatic Extraction of Information Structures from Documents. *Proc. of the First International Conference on Document Analysis and Recognition (ICDAR 91)*, Saint-Malo, Frankreich, September/Oktober 1991, Seiten 32-40.

[7] T. Bayer, H. Walischewski. Experiments on Extracting Structural Information from Paper Documents using Syntactic Pattern Analysis. *Proc. of the Third International Conference on Document Analysis and Recognition (ICDAR 95)*, Montreal, Kanada, 1995.

[8] Ottmar Lutzy. Morphic-Plus. *Ein morphologisches Analyseprogramm für die deutsche Flexionsmorphologie und Komposita-Analyse*. DFKI Dokument D-95-07, 1995.

[9] V. I. Levenshtein. *Binary Codes Capable of Correcting Deletions, Insertions, and Reversals.* Sowjet Phys. Dokl., Band 10, Nr.8, 1966, Seiten 707-710.

[10] Klaus-Peter Gores, Rainer Bleisinger. *Ein Modell zur Repräsentation von Nachrichtentypen*. DFKI Dokument D-92-28, 1992.

Entwurf einer Patternbeschreibungssprache für die Informationsextraktion in der Dokumentanalyse

Claudia Wenzel, Markus Junker

D-97-04
Documen