TIGERSearch – Ein Suchwerkzeug für Baumbanken

Wolfgang Lezius
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
http://www.ims.uni-stuttgart.de/projekte/TIGER

Zusammenfassung

In diesem Papier stellen wir die Konzeption und Implementation der Software TIGERSearch vor, die ein Suchwerkzeug für Baumbanken zur Verfügung stellt. Die Software ist für Forschungszwecke frei verfügbar und kann über die WWW-Seite des Projekts bezogen werden.

1 Einführung

Baumbanken als spezielle Form von Textkorpora sind ein fester Bestandteil der Computerlinguistik geworden. Zu ihren Anwendungsgebieten zählen die Entwicklung von Parsern, die Extraktion lexikalischer Information und phänomenbasiertes Retrieval (Lezius, 2001). Die bekanntesten englischsprachigen Baumbanken sind die *Penn Treebank* (Taylor et al., 2000) und das *Susanne Corpus* (Sampson, 1995).

Für das Deutsche stehen gegenwärtig zwei Baumbanken zur Verfügung. Die *VerbMobil-Baumbank* für gesprochene Sprache umfasst ca. 250.000 Token (Hinrichs et al., 2000). Hier wurden Dialoge zu Terminvereinbarungen transkribiert und syntaktisch annotiert. Das *Negra-Korpus* ist an der Universität des Saarlandes entstanden (Brants et al., 1999). Es besteht aus 20.000 annotierten Zeitungstext-Sätzen (ca. 350.000 Token) und ist für Forschungszwecke frei verfügbar.

Das *TIGER-Projekt* setzt auf den Ergebnissen des Negra-Projekts auf. Ziele des Projekts sind die Verfeinerung der Negra-Annotationsrichtlinien und die Annotation von weiteren 40.000 Zeitungstext-Sätzen (Brants et al., 2002). Einen Eindruck von der Korpusannotation vermittelt der TIGER-Korpus-Sampler, der 250 annotierte Sätze umfasst (Smith, 2002).

Die Menge an Information, die in einer solchen Baumbank annotiert ist, ist ohne speziell entwickelte Suchwerkzeuge nicht verwertbar. Zur Nutzbarmachung der TIGER-Baumbank ist deshalb in einem Teilprojekt das Anfragewerkzeug *TIGERSearch* entwickelt worden.

Das vorliegende Papier beschreibt zentrale Aspekte der Konzeption und Implementation des TIGERSearch-Werkzeugs. Der nachfolgende zweite Abschnitt dieses Papiers stellt die TIGER-Baumbank und damit das vom Suchwerkzeug zu unterstützende Datenmodell vor. Der dritte Abschnitt skizziert eine Korpusbeschreibungssprache, die zur Definition und Abfrage von Baumbanken verwendet wird. Das Verarbeitungskonzept für Korpusanfragen und dessen effiziente Umsetzung wird im fünften Abschnitt beschrieben. Der sechste Abschnitt zeigt abschließend Aspekte der Benutzeroberfläche des TIGERSearch-Werkzeugs auf. Eine vollständige Beschreibung des TIGERSearch-Projekts ist in (Lezius, 2002) zu finden.

2 Die TIGER-Baumbank

Im TIGER-Korpus sind Prädikat-Argument-Strukturen als Teile eines ungeordneten Baums, d.h. eines Graphen mit kreuzenden Kanten, annotiert. Für eine linguistische Begründung dieser Vorgehensweise sei auf (Brants et al., 1999) verwiesen.

Die verwendete Annotation unterscheidet drei Ebenen (vgl. Abb. 1): a) Tags auf Wortebene, die die Wortart und morphosyntaktische Information angeben (z.B. `NN` für Nomen, `Masc.Nom.Sg` für Maskulinum Nominativ Singular); b) Knotenbeschriftungen für phrasale syntaktische Kategorien (z.B. `NP` für Nominalphrase); c) Kantenbeschriftungen für syntaktische Funktionen (z.B. `HD` für Kopf, `SB` für Subjekt). Eine Lemma-Annotation ist in Vorbereitung.

Kreuzende Kanten werden neben der Repräsentation der Prädikat-Argument-Struktur auch zur Kodierung linguistischer Abhängigkei-

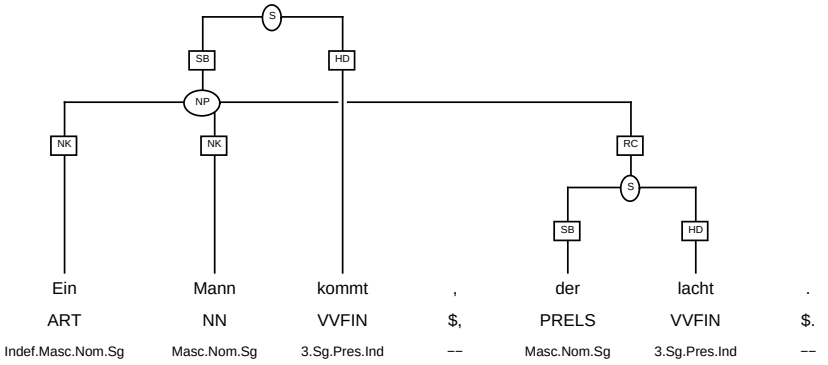

Abbildung 1: Annotation von Extraposition durch kreuzende Kanten

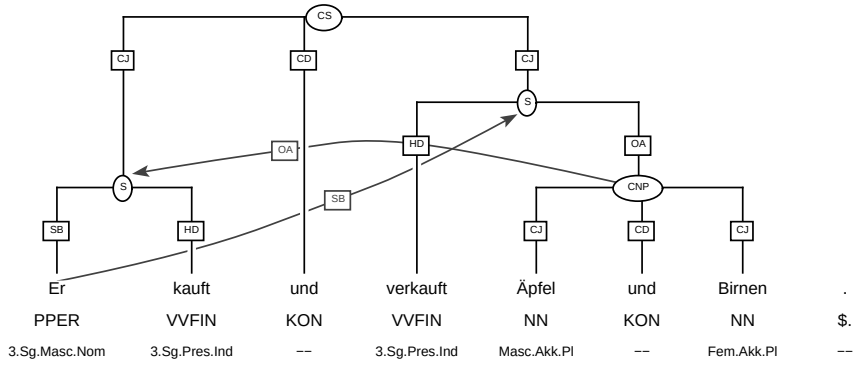

Abbildung 2: Annotation von Koordination durch sekundäre Kanten

ten verwendet. In Abb. 1 liegt beispielsweise der extraponierte Relativsatz *der lacht* vor, der über eine kreuzende Kante mit der Nominalphrase des Bezugsnomens *Mann* verknüpft wird.

Eine weitere Besonderheit des Datenmodells stellen die so genannten sekundären Kanten dar. Sie werden u.a. bei der Behandlung der Koordination eingesetzt (vgl. Abb. 2). Die koordinierten Elemente werden hier zusammengefasst, um eine neue Konstituente zu bilden (hier: `CNP` für koordinierte Nominalphrasen und `CS` für koordinierter Satz). Dabei kann es vorkommen, dass Konstituenten in einem der Konjunkte fehlen. In diesem Fall werden sekundäre Kanten verwendet. Im vorliegenden Beispiel ist das Personalpronomen *Er* Subjekt des ersten und zweiten Konjunkts; *Äpfel und Birnen* entsprechend das Akkusativobjekt beider Konjunkte.

Die beschriebene Annotation kann als hybrider Ansatz verstanden werden, der Konstituentenstrukturen mit funktionalen Abhängigkeiten verbindet, wie sie Dependenzstrukturen ausdrücken. Die in diesem Datenmodell verwendeten Strukturen werden auch als schwache Graphstrukturen bezeichnet, da sie nur in geringem Umfang von der Ausdruckskraft eines Graphen Gebrauch machen, aber dennoch mächtiger als Baumstrukturen sind. Das Datenmodell wird in (Lezius, 2002) formal definiert.

3 Verwandte Arbeiten

Für Korpora, die auf Tokenebene annotiert wurden (mit Attributen wie Wortform, Lemma, Wortart usw.), stehen bereits zahlreiche Suchwerkzeuge zur Verfügung. Einer der bekanntesten Vertreter ist das zur IMS Corpus Workbench gehörende Programm *CQP* (Evert, 2001). CQP unterstützt mit Einschränkungen auch syntaktische Annotation. Die Ausdruckskraft reicht aber für komplexere Anfragen nicht aus (Lezius, 2002).

Für Baumbanken sind bislang nur wenige Suchwerkzeuge entwickelt worden. Das erste verfügbare Werkzeug war *tgrep*, das mit der ersten Version der Penn Treebank ausgeliefert wurde (Pito, 1993). Es ist auf die Erfordernis-

se der Penn Treebank zugeschnitten und verarbeitet daher lediglich Korpora, die in Form von Baumstrukturen annotiert worden sind. Zur Beschleunigung der Anfrageverarbeitung wird ein Index verwendet. Durch die fehlende grafische Benutzeroberfläche und eine schwer zu erlernende Anfragesprache ist tgrep für den Gelegenheitsbenutzer aber nicht geeignet.

Einen anderen Weg verfolgt die Software *ICECUP* (Wallis und Nelson, 2000). Sie basiert vollständig auf einer grafischen Benutzeroberfläche, die sowohl die grafische Eingabe von Anfragen als auch die grafische Visualisierung von Anfrageergebnissen erlaubt. Durch diese Konzeption erweitert sich der Adressatenkreis des Werkzeugs, das auch für Anwender geeignet ist, die nur wenig Erfahrung im Umgang mit Anfragewerkzeugen haben. Das indexbasierte Werkzeug ist genau wie tgrep auf die Annotation in Form von Baumstrukturen eingeschränkt.

Parallel zu TIGERSearch ist das Werkzeug *VIQTORYA* zur Suche auf der VerbMobil-Baumbank entwickelt worden. VIQTORYA unterstützt wie TIGERSearch (abgesehen von sekundären Kanten) das Graphenmodell des Negra-Formats und bildet eine in diesem Format vorliegende Baumbank auf eine relationale Datenbank ab (Kallmeyer, 2000). Anfragen in einer speziell entwickelten Anfragesprache werden in SQL-Anfragen übersetzt und auf der Datenbank ausgewertet. Bei dieser Architektur wird die Entwicklung einer Anfrageverarbeitungskomponente eingespart. Problematisch ist die speicherintensive Abbildung der Graphstrukturen auf die relationale Datenbank. Hier bleibt abzuwarten, ob auch größere Korpora auf diese Weise verarbeitet werden können.

Die Aufgabe des TIGERSearch-Projekts bestand darin, die Erfahrungen der genannten Projekte zu nutzen und auf das beschriebene Graphen-Datenmodell anzuwenden. Als Ziel des Projekts wurde die Entwicklung eines Suchwerkzeugs festgelegt, das Baumbanken unterstüzt, deren Annotation wahlweise auf Baumstrukturen oder auf den beschriebenen Graphstrukturen basiert und beliebige Attribute umfassen kann. Neben einer ausdrucksstarken und leicht erlernbaren Anfragesprache sowie einer effizienten Anfrageverarbeitung wurde als weitere Anforderung die Entwicklung einer intuitiven Benutzeroberfläche formuliert.

4 Eine Korpusbeschreibungssprache

Ein Baumbank-Anfragewerkzeug definiert zur Kommunikation mit dem Anwender in der Regel zwei Sprachen: Die *Korpusdefinitionssprache* legt das Format fest, in dem das Korpus zur Aufbereitung an das Werkzeug gereicht wird. Bei Systemen wie tgrep, die zur Suche auf englischsprachigen Baumbanken entwickelt wurden, wird meist das Klammerstrukturformat verwendet. Die Mächtigkeit reicht aber zur Repräsentation von Graphstrukturen nicht aus. In der *Korpusanfragesprache* werden die Korpusanfragen formuliert. Korpusdefinitionssprache und Anfragesprache unterscheiden sich in den verfügbaren System deutlich.

Im TIGERSearch-Projekt ist versucht worden, die Trennung zwischen Korpusdefinition und Korpusanfrage aufzuheben. Als Vorteil muss lediglich ein Parser für beide Formate implementiert werden, und auch der Anwender muss sich mit nur einem Format auseinander setzen. Daneben ist das Design der beiden Formate als Einheit in sich geschlossener, da die Korpusdefinition eine echte Teilmenge der Korpusanfrage darstellt. Der Unterschied besteht lediglich im Grad der Spezifikation (vollständig spezifiziert vs. unterspezifiziert). Auch der formale Nachweis der Korrektheit und Vollständigkeit des Suchwerkzeugs wird erleichtert, da die formale Semantik nur einer Sprache berücksichtigt werden muss.

4.1 Die Beschreibungssprache

An dieser Stelle werden die Grundzüge der Korpusbeschreibungssprache skizziert. Eine vollständige Beschreibung der Konzeption der Sprache findet sich in (Lezius, 2002).

Da das Datenmodell Graphstrukturen vorsieht, orientiert sich die Beschreibungssprache an Begrifflichkeiten und Sichtweisen der Graphentheorie. Die Basisebene der Sprache ist daher die Ebene der Knoten. Auf der *Knotenebene* werden Knoten durch boolesche Ausdrücke über Attribut-Wert-Paaren beschrieben. Die Darstellungsweise orientiert sich dabei an der Notation von Merkmalsstrukturen. Das folgende Beispiel beschreibt ein Personalpronomen:

```
[pos="PPER"]
```

Auf der *Knotenrelationsebene* werden Knoten durch Relationen miteinander verknüpft. Für

Graphen stehen die Basisrelationen direkte Dominanz (>) und lineare Präzedenz (.) sowie abgeleitete Relationen wie die allgemeine Dominanz (>*) oder die Geschwisterbeziehung ($) zur Verfügung. Für die Abfrage einer Kantenbeschriftung L ist eine direkte Dominanzbeziehung mit Parameter (>L) eingeführt worden. Der folgende Ausdruck beschreibt einen Satzknoten, der ein Personalpronomen dominiert (vgl. den linken Teilbaum in Abbildung 2). Dabei bildet das Pronomen das Subjekt des Teilsatzes (vgl. Kantenbeschriftung SB).

```
[cat="S"] >SB [pos="PPER"]
```

Auf der *Graphbeschreibungsebene* schließlich werden Knotenrelationen durch boolesche Ausdrücke verknüpft. Um die Identität zweier Knoten auszudrücken, werden Variablen eingesetzt:

```
(#v:[cat="S"] > [pos="PPER"]) &
(#v > [pos="VVFIN"])
```

Daneben können Prädikate wie `discontinuous(#v)` (die Phrase enthält eine kreuzende Kante), `arity(#v,num)` (die Phrase enthält *num* Töchterknoten) oder `root(#v)` (die Phrase ist Wurzelknoten des Graphen) verwendet werden. Der folgende Ausdruck beschreibt extraponierte Relativsätze in der TIGER-Baumbank (vgl. Abbildung 1): Konstituenten, die einen Relativsatz bilden, werden mit der eingehenden Kante RC (*relative clause*) kodiert. Durch die Verwendung des Prädikats `discontinuous` werden nur extraponierte Relativsätze zugelassen:

```
(#N:[cat="NP"] >RC [cat="S"]) &
discontinuous(#N)
```

4.2 Korpusdefinition vs. Korpusanfrage

Jeder Ausdruck der Korpusbeschreibungssprache stellt eine Korpusanfrage dar. Eine Korpusdefinition hingegen unterliegt strengen Einschränkungen. Sie erfolgt graphenweise und legt dabei jeden Korpusgraphen in eindeutiger Weise fest: Alle Attributwerte werden spezifiziert, die Beziehungen zwischen den Knoten exakt definiert. Als einzige logische Verknüpfung ist die Konjunktion erlaubt. Das folgende Beispiel definiert den Teilsatz *Er kauft* des zweiten Beispielgraphen[1] in Abb. 2. Eine formale Definition der Syntax und Semantik der Korpusbeschreibungssprache ist in (Lezius, 2002) zu finden.

```
<sentence id="graphid">
  "t1":[word="Er" & pos="PPER"] &
  "t2":[word="kauft" & pos="VVFIN"] &
  "n1":[cat="S"] &
  "t1" . "t2"  &
  "n1" >SB "t1"  &  "n1" >HD "t2" &
  arity("n1",2) & root("n1")
</sentence>
```

Die Ausdruckskraft der Anfragesprache ist mit der des VIQTORYA-Systems vergleichbar. Es steht zwar in der vorliegenden Sprache eine größere Palette von Knotenrelationen zur Verfügung, doch könnten die fehlenden Relationen im VIQTORYA-System mit geringem Aufwand nachgetragen werden. In beiden Anfragesprachen fehlt der Allquantor als ausdrucksstarkes Sprachmittel. Doch sollte neben der deutlich komplexeren Anfrageauswertung zunächst diskutiert werden, wie dieses Sprachmittel in die beiden Sprachen integriert werden kann, ohne dass Suchanfragen unübersichtlich werden.

Die Erlernbarkeit der Anfragesprache ist nur schwer zu beurteilen, da sie zu sehr vom Anwendertyp abhängt. Um denjenigen Anwendern entgegen zu kommen, die nur selten mit formalen Anfragesprachen arbeiten, ist eine Komponente zur grafischen Anfrageeingabe entwickelt worden (vgl. Abschnitt 6).

4.3 TIGER-XML

Die Verschmelzung der Teilsprachen zu einer gemeinsamen Korpusbeschreibungssprache schafft zwar konzeptionelle Vorteile. Beim praktischen Einsatz der Korpusdefinitionssprache ergaben sich jedoch einige technische Probleme. So können Zeichen außerhalb des System-Zeichensatzes sowie Markup-Symbole (z.B. `[word="""]`) nur über Ersatzkodierungen (z.B. `[word="\""]`) dargestellt werden. Ein weiteres Problem besteht in der fehlenden Validierung von Korpora, die dem Korpusdefinitionsformat folgen. Daneben erlaubt das Format zwar die Kodierung eines Korpus, aber nicht die Repräsentation von Ergebnissen einer Korpusanfrage zusammen mit den Korpusdaten.

Aus diesen Gründen ist ein zum Korpusdefinitionsformat äquivalentes XML-basiertes Definitionsformat entwickelt worden, das als TIGER-XML bezeichnet wird. Dieses Format löst nicht

[1] Die morphologische Annotation bleibt aus Darstellungsgründen im Folgenden unberücksichtigt.

nur die zuvor beschriebenen technischen Probleme, sondern eröffnet durch die Verknüpfung mit XSLT-Stylesheets zudem interessante Möglichkeiten bei der Konvertierung des Korpus in andere Formate. Das folgende Beispiel zeigt die TIGER-XML-Darstellung des obigen Beispielsatzes *Er kauft*. Charakteristisch für das TIGER-XML-Format ist die Repräsentation der Graphstruktur durch Referenzen. Eine Elementeinbettung kann nicht verwendet werden, da bedingt durch kreuzende Kanten die Anordnung der Terminalknoten verloren ginge.

```
<s id="graphid">
 <graph root="n1">
  <terminals>
   <t id="t1" word="Er"    pos="PPER" />
   <t id="t2" word="kauft" pos="VVFIN"/>
  </terminals>
  <nonterminals>
   <nt id="n1" cat="S">
     <edge label="SB" idref="t1" />
     <edge label="HD" idref="t2" />
   </nt>
  </nonterminals>
 </graph>
</s>
```

4.4 Architektur Korpusaufbereitung

Die Systemarchitektur der TIGERSearch-Software ist zweigeteilt (vgl. Abb. 3). Das Werkzeug *TIGERRegistry* ist für die Aufbereitung und Verwaltung der Korpora zuständig, *TIGERSearch* dient der Auswertung von Suchanfragen und Visualisierung der Anfrageergebnisse.

Die Verwendung des TIGER-XML-Formats zur Korpusdefinition führt zu folgender Architektur des TIGERRegistry-Werkzeugs: TIGER-XML stellt das zentrale Eingangsformat dar, alle zu verarbeitenden Korpora müssen zunächst in dieses Format konvertiert werden. Diese Festlegung vereinfacht die anschließende Korpusindexierung, die damit auf einem fest definierten Format erfolgt.

Die Korpusindexierung überführt die textuelle Darstellung des Korpus in eine binäre Repräsentation. Dabei wird neben der reinen Korpusrepräsentation Wissen über das Korpus gesammelt, das der Anfrageverarbeitung zu einer möglichst effizienten Verarbeitung von Anfragen verhilft.

Um alle gängigen Baumbankformate direkt zu unterstützen, werden einige Importfilter zur Verfügung gestellt, die diese Baumbanken in das TIGER-XML-Format konvertieren. Man beachte, dass bei dieser Konvertierung linguistische Reinterpretationen nötig sind. So können beispielsweise Spuren der Penn Treebank (Taylor et al., 2000) durch ein zusätzliches `trace`-Attribut oder auch durch sekundäre Kanten modelliert werden.

5 Verarbeitung von Korpusanfragen

5.1 Der TIGER-Kalkül

Die Verarbeitung von Korpusanfragen hat die Aufgabe, die Ableitbarkeit einer Anfrage aus einem Korpus zu überprüfen. Um eine theoretisch fundierte Implementation zu gewährleisten, ist der so genannte TIGER-Kalkül entwickelt worden. Es handelt sich hierbei um einen Logik-Kalkül, der die Ableitbarkeit einer Anfrage durch die Anwendung von syntaktischen Verarbeitungsregeln überprüft. Das eingesetzte Verarbeitungsmodell orientiert sich dabei am Resolutionskalkül für Hornklauseln in logischen Programmiersprachen. Durch den Nachweis der Korrektheit und Vollständigkeit des Kalküls steht ein solides Fundament für die Implementation der Anfrageauswertung zur Verfügung.

Die Aufgabe der Anfrageverarbeitung besteht damit darin, den TIGER-Kalkül korrekt und vollständig und dabei möglichst effizient zu implementieren. Der Kalkül ist zu komplex, um ihn an dieser Stelle darstellen zu können. Für eine ausführliche Beschreibung sei auf (Lezius, 2002) verwiesen. Stattdessen wird im Folgenden die Strategie der Kalkül-Implementation an einem Beispiel illustriert.

5.2 Anfrageverarbeitung

Bei der Anfrageverarbeitung wird die Ableitbarkeit einer Anfrage nacheinander für alle Korpusgraphen getestet. Die dabei verfolgte Konzeption soll nun an einem konkreten Beispiel illustriert werden. Gegeben ist dazu die folgende Anfrage, die gegen die beiden Korpusgraphen in Abb. 1 und Abb. 2 getestet werden soll:

```
#p:[cat="NP"|cat="S"] > #t:[pos="PPER"] &
arity(#p,2)
```

In einem ersten Schritt findet eine Normalisierung der Anfrage in eine Disjunktive Normal-

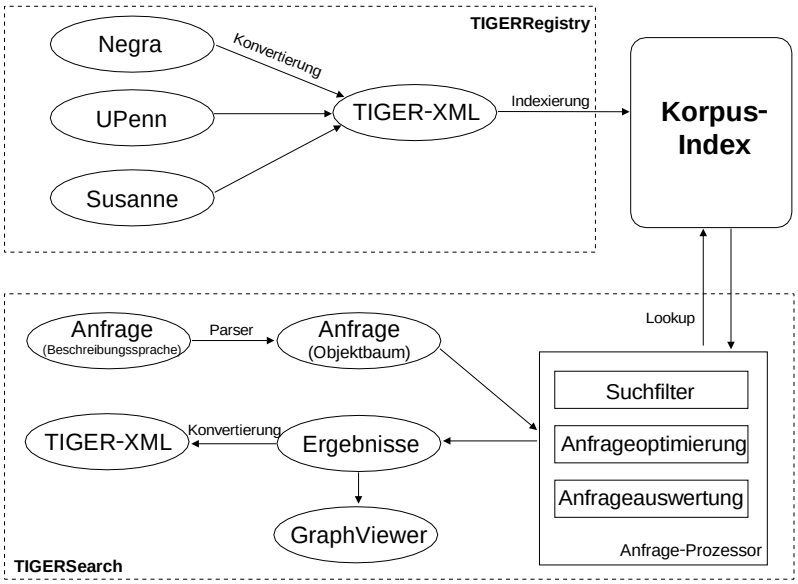

Abbildung 3: Architektur des TIGERSearch-Softwarepakets

form statt. Alle Disjunkte werden nach außen geschoben, außerdem werden Knotenbeschreibungen aus Knotenrelationen gelöst.

```
( #p:[cat="NP"] & #t:[pos="PPER"] &
  #p > #t & arity(#p,2) )  |
( #p:[cat="S"] & #t:[pos="PPER"] &
  #p > #t & arity(#p,2) )
```

Als Resultat entsteht eine Liste von Disjunkten, in der jedes Disjunkt ausschließlich konjunktive Verknüpfungen umfasst. Alle Variablen eines Disjunkts stehen damit in einem rein konjunktiven Kontext, weshalb die Disjunkte im weiteren Verlauf unabhängig von den übrigen Disjunkten verarbeitet werden können. Im Folgenden wird die Verarbeitung des zweiten Disjunkts illustriert.

Im nächsten Schritt werden die Knotenbeschreibungen aus der Anfrage herausgelöst und in die so genannte Halde eingewiesen. Die Halde verwaltet die Belegungen aller Variablen und die Bedingungen, die die Anfrage an sie geknüpft hat.[2] Die Anfrage besteht nach dem Aufbau der Halde lediglich aus einem Skelett, das keine Knotenbeschreibungen mehr enthält.

[2]Die Darstellung des Haldenmodells ist an dieser Stelle vereinfacht worden. Da die Korpusbeschreibungssprache neben Knotenvariablen zwei weitere Variablenebenen vorsieht, ist die tatsächliche Implementation deutlich komplexer (Lezius, 2002).

`#p & #t & #p > #t & arity(#p,2)`

Variable	Belegung	Bedingung
`#p`	--	`[cat="S"]`
`#t`	--	`[pos="PPER"]`

Die Vorbereitungen sind damit abgeschlossen und die eigentliche Anfrageauswertung kann beginnen. Alle Korpusgraphen werden nun nacheinander gegen die Anfrage (bestehend aus Anfrageskelett und Halde) getestet. Für jeden Korpusgraphen werden die Konjunkte des Anfrageskeletts von links nach rechts durchwandert. Dabei werden Knotenvariablen an Korpusknoten gebunden und Knotenrelationen bzw. Knotenprädikate getestet. Kann eine Variable nicht gebunden werden bzw. ist ein Test erfolglos, so wird ein Backtracking ausgelöst und die Verarbeitung wird mit dem letzten Konjunkt fortgesetzt.

Zur Illustration wird der zweite Beispielgraph aus Abb. 2 verwendet. Wird die Knotenvariable `#p` beispielsweise an den Wurzelknoten gebunden, so passt die Attributbelegung des `CS`-Wurzelknotens nicht zur Bedingung `[cat="S"]`, die in der Halde für `#p` festgehalten ist. D.h. die Unifikation des Korpusknotens mit der Knotenbeschreibung `#p` ist erfolglos. Wird aber `#p` im weiteren Verlauf an den `S`-Knoten des linken Teilsatzes gebunden, so ist die Unifikation erfolgreich. Das Ergebnis der Unifikation zeigt sich im veränderten Status der Halde:

```
#p & [#t] & #p > #t & arity(#p,2)
```

#p	"n1"	[cat="S"]
#t	--	[pos="PPER"]

Der Algorithmus sucht nun eine Belegung für die Variable `#t`. Der Test für das Token *Er* ist erfolgreich. Man beachte, dass `#t` nun an das Ergebnis der Unifikation der bisherigen Bedingung für `#t` mit dem Korpusknoten (d.h. auch an die Wortform *Er*) gebunden ist.

```
#p & #t & [#p > #t] & arity(#p,2)
```

#p	"n1"	[cat="S"]
#t	"t1"	[word="Er" & pos="PPER"]

Da der anschließende Relationstest und die Prüfung des Stelligkeitsprädikats ebenfalls erfolgreich verlaufen, ist ein Match gefunden worden. Die zugehörige Variablenbelegung ist in der Halde ablesbar (vgl. Teilsatz *Er kauft*).

```
#p & #t & #p > #t & arity(#p,2) √
```

#p	"n1"	[cat="S"]
#t	"t1"	[word="Er" & pos="PPER"]

Die Belegung der Halde zum Zeitpunkt des erfolgreichen Durchlaufs repräsentiert das Anfrageergebnis. Das Anfrageergebnis kann nun in die TIGER-XML-Darstellung der Korpusgraphen eingetragen werden (Lezius, 2002). Da ein Korpusgraph eine Anfrage durchaus mehrmals erfüllen kann, wird das Backtracking trotz erfolgreicher Suche fortgesetzt.

5.3 Effiziente Verarbeitung

Der Backtracking-Algorithmus führt für jeden Korpusgraphen eine erschöpfende Suche durch, was für alle erfolgreichen Suchen zu einer exponentiellen Laufzeit führt. Erfolglose Suchen scheitern dagegen meist an einem frühen Konjunkt, womit die Laufzeit insgesamt nur in seltenen Fällen beeinträchtigt ist. Durch Anfrageoptimierungs- und Suchfilterstrategien lässt sich die Effizienz zudem deutlich verbessern.

Als Beispiel für die Anfrageoptimierung soll das zweite Disjunkt der Beispielanfrage dienen:

```
#p:[cat="S"] & #t:[pos="PPER"] &
#p > #t & arity(#p,2)
```

Die Anfrageverarbeitung verläuft streng von links nach rechts. Das Binden der Variablen an Korpusknoten und die damit verbundene Prüfung, ob die Attributbelegungen des Korpusknoten zu den Bedingungen der Variable passen, stellt dabei das zeitintensivste Teilziel dar. Durch eine Umformulierung der Anfrage lassen sich viele unnötige Prüfungen für die Token-Variable vermeiden:

```
#p:[cat="S"] & arity(#p,2) &
#t:[pos="PPER"] & #p > #t
```

Durch diese Anfrageoptimierung wird eine Geschwindigkeitssteigerung von 75% erreicht. Für komplexere Anfragen werden entsprechend deutlichere Steigerungen erzielt.

Eine weitere Idee besteht darin, die wesentliche Information aus einer Anfrage abzulesen, um eine vereinfachte Vorauswertung der Anfrage durchzuführen. Ein Knotenfilter beispielsweise reduziert die obige Anfrage auf diejenigen Knotenbeschreibungen, die restriktive Informationen tragen:

```
#t:[pos="PPER"]
```

Für diese Reduktionen werden statistische Korpusinformationen verwendet, die vom Index zur Verfügung gestellt werden. Man beachte, dass die ermittelte Anfrage durch Entfernen von Restriktionen ermittelt wurde und damit eine Generalisierung der originalen Anfrage darstellt. Nur diejenigen Korpusgraphen, die die reduzierte Anfrage erfüllen, stellen damit Kandidaten zur Erfüllung der eigentlichen Anfrage dar. Aus dem Index sind diese Graphkandidaten unmittelbar ablesbar. Im Falle des TIGER-Korpus verbleiben im Beispiel nur noch 25% aller Korpusgraphen als Kandidaten.

5.4 Architektur Anfrageverarbeitung

Die Architektur der Anfrageverarbeitung ist in Abb. 3 dargestellt. Eine Korpusanfrage wird zunächst geparst. Nach der Anfrageoptimierung wird der Suchraum gefiltert und die Anfrageauswertung graphenweise auf den ermittelten Kandidaten durchgeführt. Dieser Algorithmus basiert auf den Indexdaten, die bei der Korpusaufbereitung ermittelt wurden. Die Anfrageergebnisse können auf der Benutzeroberfläche eingesehen oder im TIGER-XML-Format exportiert werden.

6 Aspekte der Benutzeroberfläche

Die Arbeiten am Suchwerkzeug ICECUP haben gezeigt, welche Bedeutung ein durchdachtes Konzept einer grafischen Benutzeroberfläche für die Verwendbarkeit eines Werkzeugs haben kann (Wallis und Nelson, 2000). Aus diesem Grunde wurden in TIGERSearch beide Hauptkomponenten zunächst mit einer Benutzeroberfläche ausgestattet.

Darüber hinaus wurden spezielle Ansätze zur visuellen Herangehensweise an ein Korpus entwickelt. Die Visualisierung von Anfrageergebnissen geschieht im so genannten TIGERGraph-Viewer. Er stellt für eine Suchanfrage die matchenden Korpusgraphen dar und hebt die am Match beteiligten Knoten farbig hervor (Lezius, 2002).

Eine große Hilfe für viele Anwender ist die grafische Eingabe von Benutzeranfragen (Voormann und Lezius, 2002). Hier können Anfragen durch eine Kombination der Elementarbausteine Knoten und Relationen gezeichnet werden. Der Anwender kann das Erlernen der Anfragesprache ausblenden und sich ganz auf die explorative Arbeit mit dem Korpus konzentrieren.

7 Zusammenfassung

Das vorliegende Papier beschreibt die Konzeption eines Suchwerkzeugs für Baumbanken. Das Werkzeug ist vollständig in Java implementiert und damit für alle Plattformen verfügbar.

Aktuelle Arbeiten konzentrieren sich auf die Verbesserung der Verarbeitungseffizienz durch die Implementation komplexer Suchfilter. Technische Weiterentwicklungen sehen den Aufbau eines Client-/Server-Szenarios vor, das schlanke Clients zur Eingabe von Suchanfragen und Darstellung von Suchergebnissen vorsieht und die Auswertung der Anfragen serverseitig auf einem Hochleistungsserver durchführt. Dadurch eröffnen sich interessante Möglichkeiten für die Distribution des TIGER-Korpus.

Literaturverzeichnis

Thorsten Brants, Wojciech Skut, und Hans Uszkoreit. 1999. Syntactic annotation of a German newspaper corpus. In *Proceedings of the ATALA Treebank Workshop*, S. 69–76, Paris.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, und George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol.

Stefan Evert, 2001. *CQP Query Language Tutorial.* Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Erhard W. Hinrichs, Juli Bartels, Yasuhiro Kawata, Valia Kordoni, und Heike Telljohann. 2000. The Verbmobil Treebanks. In Werner Zühlke und Ernst G. Schukat-Talamazzini, Herausgeber, *Konvens 2000 Sprachkommunikation*, S. 107–112. VDE-Verlag.

Laura Kallmeyer. 2000. A query tool for syntactically annotated corpora. In *Proceedings of Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, S. 190–198, Hongkong.

Wolfgang Lezius. 2001. Baumbanken. In Kai-Uwe Carstensen, Christian Ebert, Cornelia Endriss, Susanne Jekat, Ralf Klabunde, und Hagen Langer, Herausgeber, *Computerlinguistik und Sprachtechnologie - Eine Einführung*, S. 377–385. Spektrum Akademischer Verlag, Heidelberg, Berlin.

Wolfgang Lezius. 2002. Ein Werkzeug zur Suche auf syntaktisch annotierten Textkorpora. Dissertation, in Vorbereitung.

Richard Pito, 1993. *Documentation for tgrep.* LDC, University of Pennsylvania.

Geoffrey Sampson. 1995. *English for the Computer: The SUSANNE Corpus and Analytic Scheme.* Clarendon Press.

George Smith. 2002. A brief introduction to the TIGER Corpus Sampler. Projektbericht.

Ann Taylor, Mitchell Marcus, und Beatrice Santorini, 2000. *The Penn Treebank: An Overview.* Language and Speech series. Kluwer Academic Press.

Holger Voormann und Wolfgang Lezius. 2002. TIGERin - Grafische Eingabe von Benutzeranfragen für ein Baumbank-Anfragewerkzeug. In *Tagungsband zur Konvens 2002*, Saarbrücken.

Sean Wallis und Gerald Nelson. 2000. Exploiting fuzzy tree fragments in the investigation of parsed corpora. *Literary and Linguistic Computing*, 15(3):339–361.