Satzklammer annotieren und Tags korrigieren – Ein mehrstufiges „Top-Down-Bottom-Up"-System zur flachen, robusten Annotierung von Sätzen im Deutschen*

*Frank Henrik Müller und Tylman Ule***

Zusammenfassung

Der vorliegende Ansatz stellt ein System zum flachen, automatischen Parsen deutscher Sätze vor, das sich die Restriktionen im Aufbau der Satzklammer zu Nutze macht und auf ein gemischtes *Top-Down-Bottom-Up*-Verfahren zurückgreift. Dabei werden direkt nach der Zuweisung der Wortartenmarkierungen (Tagging) die Satzklammern erkannt und somit sowohl topologische Felder als auch Satztypen bestimmt. Erst nach diesen Schritten werden nicht-rekursive, ununterbrochene Kernphrasen (Chunks) annotiert.

Das System verwendet hintereinander geschaltete (kaskadierte) Transduktoren, die von Regeln mit der Ausdrucksstärke Regulärer Ausdrücke gesteuert werden und die jeweils auf den Strukturen der vorangehenden Transduktoren aufbauen. Das System verfährt inkrementell, ist modular aufgebaut und arbeitet robust. Es benötigt als erste Eingabe getaggten Text, wobei es jedoch falsche Tags mit denselben Restriktionen, die es für die Annotierung der Satzklammer nutzt, korrigiert, indem es Tags so ändert, dass sie sich in parsebare Abfolgen fügen.

22.1. Nutzung von Satzklammern zum Parsen

22.1.1. Satzklammern und mehrstufige Parsingverfahren

Für die Annotierung von Sätzen in deutschen Textkorpora lässt sich ein Phänomen nutzen, das u. a. als *Satzklammer* (Helbig und Buscha, 1996, Kap. 14.1) bekannt ist. Diese beruht auf dem Prinzip der „Distanzstellung der Prädikatsteile und anderer (syntaktisch eng zusammengehöriger) Elemente" (Bußmann, 1990). Sie gliedert den Satz in topologische Felder, die wenigen syntaktischen Restriktionen und einigen syntaktischen Präferenzen unterliegen, die alle zur weiteren

* Erschienen in: *Proceedings der GLDV-Frühjahrstagung 2001*, Henning Lobin (Hrsg.), Universität Gießen, 28.–30. März 2001, Seite 235–244. `http://www.uni-giessen.de/fb09/ascl/gldv2001/`

** Diese Arbeit wurde teilweise durch ein Stipendium der Deutschen Forschungsgemeinschaft (Graduierten-Kolleg Integriertes Linguistik-Studium) ermöglicht. Sie entstand im Rahmen des SFB 441 – *Linguistische Datenstrukturen* und des Projektes *Deutsches Referenzkorpus*.

Annotierung nutzbar sind. Der Aufbau der Satzklammer ist mit einer Regulären-Ausdrucks-Grammatik fassbar. Es ist daher möglich, zunächst (Teil-)Satzstrukturen zu erkennen und in Relation zueinander zu stellen, um dann *innerhalb* dieser Strukturen grammatische Beziehungen zu annotieren.

Solche und ähnliche Verfahren sind unter den Namen *divide-and-conquer strategy* (Peh und Ting, 1996) und *Parsen in zwei Schritten* (Wauschkuhn, 1996) bekannt und können allgemein als *mehrstufige Parsingverfahren* (Braun, 1999) bezeichnet werden. Für das Deutsche wurde ein solches Verfahren bereits in einem Informationserschließungs-System eingesetzt (Braun, 1999, Neumann et al., 2000). Es handelt sich hierbei um eine Technik der *Suchraumeingrenzung und -beschreibung*, da zunächst definierte Bereiche innerhalb eines Satzes erkannt und benannt werden, um danach innerhalb dieser Bereiche Strukturen zu erkennen. Diese Technik hat zum einen den Vorteil, dass die Einschränkung des Suchraums Anbindungsprobleme verringert und zudem das System effizienter macht, sie ermöglicht zum anderen aber auch, dass die grammatischen Eigenschaften der Suchräume beschrieben werden und entsprechend unterschiedliche Grammatiken angewendet werden können.

Bei dem von Abney vorgestellten Verfahren des Chunk-Parsing (Abney, 1996) handelt es sich ebenfalls um ein mehrstufiges Verfahren, da nach dem Chunking einfache Sätze erkannt werden und somit ebenfalls der Suchraum für folgende Annotationsschritte (Hinrichs et al., 2000) eingeschränkt wird (*containment of ambiguity*). Jedoch ändert sich dort die Annotationsrichtung – im Gegensatz zu dem vorliegenden Verfahren – nicht. Im vorliegenden Verfahren werden jedoch zunächst *bottom-up* komplexe Verbstrukturen und andere Teile der Satzklammer erkannt, um dann *top-down* topologische Felder und (Teil-)Satztypen zu erkennen, bevor wieder *bottom-up* Chunks erkannt werden.

22.1.2. Vorteile der vorgezogenen Satzklammer-Annotierung

Ein mehrstufiges Parsingverfahren mit kombinierten *Top-Down-* und *Bottom-Up*-Schritten, bei dem die Satzklammer zuerst erkannt wird, bietet sich aus folgenden Gründen an:

- die Satzklammer besteht aus den einfunktionalen Konstituenten des Satzes
- die Zusammensetzung der Satzklammer deckt den Satztyp auf
- die Satzklammer teilt den Satz in topologische Felder, deren Restriktionen und Präferenzen bei weiterer Annotierung helfen

Da die Konstituenten, aus denen die Satzklammer besteht, einfunktionale Konstituenten sind (z. B. Konjunktionen und Verben), kann man sie im Gegensatz zu z. B. Nominalgruppen, die u. a. Subjekt, Objekt, Apposition oder Genitivattribut sein können, direkt nach dem Tagging ihrer syntaktischen Funktion zuordnen. Es liegt daher nahe, gerade mit den Konstituenten der Satzklammer die Annotierung nach dem Tagging fortzusetzen. Wenn es gelänge, diese Konstituenten eindeutig zu erkennen, wären schon nach dem Tagging die Voraussetzungen geschaffen, die Satzklammer und damit das „Gerüst“ des Satzes zu erkennen, wie die Abb. 22.1 und 22.2 zeigen[1]. Abb. 22.2 zeigt besonders deutlich, dass allein die Klammerelemente die Grobstruktur

[1] Die Benennung der Verbal-Chunks ist nach hierarchischen und mnemotechnischen Gesichtspunkten erfolgt. Das „L“ oder „R“ an dritter Stelle steht für den linken oder rechten Klammerteil. Die folgenden aus je zwei Buchstaben bestehenden Gruppen nehmen die jeweils zweiten und dritten Buchstaben der Tags, die Verbart und Finitheit beschreiben, auf. Die Reihenfolge der Verben wird in der Reihenfolge ihrer syntaktischen Abhängigkeit angegeben.

des Satzes und die Kategorie der einzelnen Satztypen deutlich machen. So kann der im Vorfeld eingebettete Relativsatz durch Relativpronomen und rechte Verbklammer erkannt und begrenzt werden.[2] Das Vorfeld lässt sich durch den Satzanfang und die linke Klammer begrenzen. Somit wird durch die Anordnung der Klammerelemente klar, dass es sich bei dem vorliegenden Satz um einen Aussagesatz mit im Vorfeld eingebettetem Relativsatz handelt.

```
{VF
     [NC
          [ART Der]
          [NN Vorstand]  ]
     [NC
          [ART der]
          [NN Unionsfraktion]  ]
}
[VCLAF
     [VAFIN wird]  ]         <-- Linker Teil der Satzklammer
{MF
     [NC
          [PRF sich]  ]
     [PC
          [APPR an]
          [NC
               [PDAT diesem]
               [NN Dienstag]  ]  ]
     [AVC
          [ADV noch]
          [ADV einmal]  ]
}
[VCRVI
     [VVINF treffen]  ]      <-- Rechter Teil der Satzklammer
[$. .]
```

Abbildung 22.1.: Einfacher Satz

Abb. 22.1 zeigt einen der Vorteile der vorgezogenen Feldererkennung. So besagt eine syntaktische Restriktion für das Deutsche, dass lediglich *eine* Konstituente im Vorfeld vorkommen darf. Aus dieser Feststellung folgt, dass – anders als beispielsweise in *Dies sagt der Vorstand der Unionsfraktion. – der Unionsfraktion* in Abb. 22.1 nicht ambig, sondern eindeutig als Genitivattribut zu erkennen ist. Es können also spezielle Grammatiken für die verschiedenen Felder angewendet werden. Eine andere Restriktion besagt, dass Koordination nicht über Feldgrenzen hinweg geschehen darf, was die Analyse koordinierter Strukturen wesentlich vereinfacht. Bei den anderen in der Feldertheorie beschriebenen nutzbaren Regelmäßigkeiten handelt es sich im Wesentlichen um syntaktische Präferenzen. So gelten im Mittelfeld einige syntaktische Präferenzen, die sowohl sehr allgemein (Weinrich, 1993) als auch sehr ausführlich (Griesbach, 1986) beschrieben werden können. Es bedarf weiterer Untersuchungen, um diese Präferenzen für die syntaktische Annotierung nutzbar zu machen.

[2] Ausklammerungen im Nachfeld werden durch den Beginn von neuen Strukturen erkannt und begrenzt.

22.2. Systemaufbau

22.2.1. Voraussetzungen, Werkzeuge und Formate

Das System benötigt als Eingabe getaggten Text (Tagger: Brants, 1998; Tagset: Schiller et al., 1995) und erzeugt als Ausgabe gechunkten und mit Satzklammern und Satztypen versehenen Text. Der Text aller Verarbeitungsstufen ist in XML kodiert (Bray et al., 2000). Das System der kaskadierten Transduktoren arbeitet mit den Text-Tokenisation-Tools (TTT) der Language Technology Group (LTG) Edinburgh (Grover et al., 1999). Mit dessen Hauptkomponente *fsgmatch*, einem Transduktor, lassen sich XML-Elemente matchen und verändern. So lässt sich der getaggte Eingabetext durch eine Kette von Transduktoren Schritt für Schritt mit komplexeren Informationen anreichern. Das Ausgabeformat orientiert sich eng am CES (Ide, 2000), ist aber im Bereich der morphosyntaktischen Annotation erweitert.

Das XML-Format hat sich im Bereich der linguistischen Annotierung als sehr nützlich erwiesen. Mit den TTT werden linguistische Konstituenten in XML-Elemente eingefasst und auf diese Weise mit Attribut-Wert-Paaren (Merkmalsstrukturen) versehen. Die Erweiterbarkeit von XML eignet sich dabei besonders für die Abbildung komplexer linguistischer Datenstrukturen. So können die Module für unterschiedliche Annotierungsebenen unter dem Dach eines einheitlichen Formates arbeiten und die Herkunft von verschiedenen Annotationsdaten für eine Ebene (beispielsweise bei der Verwendung verschiedener Tagger) in der XML-Merkmalsstruktur integriert und individuell kenntlich gemacht werden. Ein weiterer Vorteil des XML-Formates besteht in der stets wachsenden Zahl der für die Verarbeitung von XML-Dokumenten verfügbaren Werkzeuge.

22.2.2. Auswirkung der Tagging-Fehler auf die weitere Verarbeitung

Bereits in den ersten Stufen der Kaskade wird deutlich, wie wichtig es ist, die Satzklammer zuerst zu annotieren. Wenn sich die weitere Annotierung von Satzklammer und Chunks auf die Güte der morphosyntaktischen Annotation verlässt, dann verhindern oft schon wenige Tagging-Fehler die richtige Annotierung ganzer Sätze. Da aber aktuelle Tagger lediglich über eine Genauigkeit von 96%–98% verfügen, die bei unbekannten Textsorten noch nachlässt, muss eine Fehlerbehandlung in die weitere Verarbeitung mit einbezogen werden. Eine mögliche Strategie ist es, das Tagging zu verbessern, indem man Tagfolgen durch einen Parser abgleicht und unterspezifizierte oder falsche Tags revidiert (Rösner, 2000, Hirakawa et al., 2000). Eine solche Strategie wird auch im vorliegenden System verfolgt. Dabei versucht das System, Sätze nach der Theorie der topologischen Felder zu erkennen[3] (Weinrich, 1993). Fügt sich eine Struktur nicht in eine mögliche Sequenz, so wird die nicht-passende Konstituente umannotiert, soweit dies hinsichtlich ihrer Ambiguitätsklasse in Frage kommt (vgl. Abb. 22.3).

Dabei ist es jedoch wichtig, die Schwere der Tagging-Fehler und die Möglichkeit ihrer Korrektur zu berücksichtigen. So wird bei der Evaluation von Taggern oft lediglich eine einzige Fehlerquote angegeben. Die Fehler sind jedoch von höchst unterschiedlicher Natur. Es ist zu unterscheiden zwischen Fehlern, die im Wesentlichen auf die Tag-Ebene beschränkt bleiben, Fehlern, die im Wesentlichen die Chunk-Ebene beeinträchtigen und Fehlern, die die Erkennung der Satzstruktur unmöglich machen. Zur ersten Kategorie gehört z. B. der Fehler, der auf die falsche Erkennung des Paares Appellativum ↔ Eigenname (NN ↔ NE) zurückzuführen ist. Dieser Fehler machte beim Taggen des Negra-Korpus (176 371 Token) mit dem auf verschiedenen

[3] Mehrfach eingebettete Sätze werden durch mehrmaliges Anwenden der Satzgrammatik verarbeitet.

```
{VF
     [NC
          [ART Eine]
          [NN Reihe]  ]
     [PC
          [APPR von]
          [NC
               [NN Menschen]  ]  ]
     [$, ,]
     [REL
          {CF
               [PRELS die]              <-- Linker Teil der Satzklammer
          }
          {MF
               [NC
                    [NN Zeugen]  ]
               [NC
                    [PDAT dieser]
                    [AJCatt
                         [ADJA traurigen]  ]
                    [NN Begebenheit]  ]
          }
          [VCRAF
               [VAFIN waren]  ]         <-- Rechter Teil der Satzklammer
     ]
     [$, ,]
}
[VCLAF
     [VAFIN ist]  ]                     <-- Linker Teil der Satzklammer
{MF
     [PC
          [PROAV seitdem]  ]
}
[VCRVP                                  <-- Rechter Teil der Satzklammer
     [VVPP verschwunden]  ]
[$. .]
```

Abbildung 22.2.: Im Vorfeld eingebetteter Relativsatz

Textsorten trainiertem tnt-Tagger ca. ein Viertel (= 25,46%) aller Fehler aus.[4] Die weitere Analyse behindert dieser Fehler jedoch kaum. Zur zweiten Kategorie gehören Fehler, die – wie das Paar Artikel ↔ substituierendes Demonstrativpronomen (ART ↔ PDS) – Fehler in der Chunk-Analyse hervorrufen, die die Erkennung von Satzklammern und Satztypen aber nicht behindern. Diese Gruppe umfasst viele Fehlerpaare und ist schwer zu umreißen. Zur dritten Kategorie gehört schließlich die Gruppe finites Verb ↔ nicht-finites Verb (z. B. VVFIN ↔ VVINF/VVPP), die fast jeden fünften Fehler ausmacht (19,85%). Diese Gruppe – zusammen mit allen anderen Fehlerarten, die die Satzklammer betreffen – verhindert eine Annotierung auf Satzebene.

Eine genauere Untersuchung der einzelnen Fehlertypen und ihrer Auswirkung auf den weiteren Annotierungsprozess steht noch aus. Es muss jedoch festgehalten werden, dass zwischen ihnen im Hinblick auf die weitere Analyse erhebliche Unterschiede bestehen, wobei insbesondere Fehler der dritten Kategorie *vor* der Erkennung von Satzstrukturen behoben werden müssen, wenn man nicht ein Scheitern der Annotierung ganzer Sätze in Kauf nehmen will. Damit muss Abschied genommen werden von einem reinen *Bottom-Up-* oder *Top-Down-*Ansatz, da Chunks weiterhin *bottom-up* erkannt werden. Würden jedoch Chunks vor der Satzstruktur erkannt, würden wegen fehlerhafter Tags viele Konstituenten der Satzklammer in Chunks integriert, was eine spätere Satzstrukturerkennung unmöglich machte. Dies trifft insbesondere auf Strukturen wie in Abbildung 22.2 zu. Hier bilden die Konstituenten *die* und *waren* die Satzklammer. Wird *die* – was ein häufiger Tagging-Fehler ist – fälschlicherweise als Artikel erkannt, wird es mit dem folgenden Nomen zu einem Nominal-Chunk zusammengefasst. Die Satzklammer kann dann nicht mehr erkannt werden und die Erkennung der gesamten Satzstruktur schlägt fehl.

Hierbei ist auch ein weiteres wichtiges Phänomen zu beachten: Da die Elemente der Satzklammer trotz enger syntaktischer Zusammengehörigkeit große Distanz aufweisen, werden sie von einem statistischen Tagger, der üblicherweise den engeren Kontext[5] verwendet, oft falsch annotiert. Die Tab. 22.1 und 22.2 (S. 241) zeigen für beide Sprachmodelle die überdurchnittliche Fehlannotierung von Verben. Die Fehlerquoten für Konjunktionen und Relativpronomina liegen beim tnt-Tagger-sm2 ebenfalls über dem Durchschnitt von 4,44% (5,34% bzw. 9,96%).[6] Dabei fällt besonders ins Gewicht, wenn ein ambiges Token in einer Ambiguitätsklasse mit einem hochfrequenten Token ist, wie dies bei den Relativpronomina und den Artikeln aber auch bei Konjunktionen, die in einer Ambiguitätsklasse mit Präpositionen oder Adverbien sind, der Fall ist. Das hier vorgestellte System versucht, diese durch statistische Tagger entstehenden Ungleichgewichte zu verkleinern.

22.2.3. Tags korrigieren und Satzklammer annotieren

Da Tagger den weiteren Kontext üblicherweise nicht mit einbeziehen, und gerade im Deutschen mit der Satzklammer syntaktisch zusammengehörige Elemente, die für die Satzstrukturerkennung zentral sind, weit auseinander stehen, liegt es nahe, die entsprechenden Tags zusammen mit der Annotierung der Satzklammer zu überprüfen und gegebenenfalls zu korrigieren, zumal sie über-

[4] Der mit diesem Sprachmodell benutzte Tagger wird fortan tnt-Tagger-sm2 genannt. Der mit dem mitgelieferten, aus den Daten des Frankfurter-Rundschau-Korpus (FR) erstellten Modell benutzte Tagger wird tnt-Tagger-sm1 genannt. Die Experimente werden auf dem Negra-Korpus durchgeführt, das aus Texten der FR besteht (Skut et al., 1998).

[5] Im Gegensatz dazu liegt die Fehlerquote von Token, bei denen sich die syntaktische Zusammengehörigkeit durch enge Abfolge ausdrückt, niedriger (beispielsweise Präpositionen: 0,87%, Artikel: 0,53% für tnt-Tagger-sm2.).

[6] Hierbei ist noch zu beachten, dass hochfrequente Konjunktionen wie *dass(daß)*, *weil* und *wenn*, die zu dieser Tag-Klasse gehören, nicht ambig sind. Die Fehlerquote für ambige Token derselben Klasse liegt folglich höher.

	unverbessert		verbessert	
Tag	Fehlerzahl	Fehlerquote	Fehlerzahl	Fehlerquote
VVFIN	924	12,36%	600	8,00%
VVPP	223	7,45%	186	6,22%
VVINF	233	9,09%	196	7,64%
VAFIN	85	1,79%	37	0,78%
VAINF	24	4,62%	7	1,35%
VMFIN	42	2,58%	21	1,29%
VMINF	23	25,00%	8	8,70%
Summe Fehler Verben	1554	7,75%	1055	5,26%
Summe Fehler Gesamt	7827	4,44%	6772	3,84%

Tabelle 22.1.: Tagging-Fehler tnt-Tagger-sm2

proportional oft falsch annotiert sind. Denn, wenn zum vollständig korrekten Taggen Teile des Parsings bereits vorweggenommen werden müssten, würde letztendlich ein Arbeitsschritt zweimal durchlaufen. Es ist somit folgerichtig, die Tag-Korrektur in das flache Parsing einzubauen.

Abb. 22.3 (S. 243) veranschaulicht den Vorgang der Tag-Korrektur. In dieser Abbildung wird vom ungünstigsten Fall ausgegangen, dass beide Seiten der Satzklammer falsch annotiert sind. Das System sucht also nach einer „Most Preferable Parsable POS-Sequence“ (Hirakawa et al., 2000) und korrigiert Tags von nicht parsebaren Sequenzen.

Zunächst versucht eine Regel – davon ausgehend, dass der Tagger richtig annotiert hat –, zu dem als Infinitiv-Chunk und rechter Satzklammer annotierten Chunk (VCRMIVI) einen entsprechenden finiten Chunk (gesucht: VCLAF) zu matchen, wie er nach den syntaktischen Regeln vorhanden sein müsste. Da kein entsprechender finiter Chunk vorliegt, korrigiert das System die Tagger-Entscheidung, indem es einen finiten Chunk (VCRMFVI) annimmt und gleichzeitig das Tag ändert (VMINF → VMFIN). Den finiten Chunk der rechten Klammer versucht es mit einer subordinierenden Konjunktion oder einem Relativpronomen zu matchen, da die Struktur nun auf einen eingeleiteten Nebensatz hindeutet. Da kein Subordinator vorliegt, untersucht das System Token, die von der Form und der Distribution her in das Muster passen und annotiert sie um. Danach können Satzklammer gematcht und topologische Felder annotiert werden.

Für den tnt-Tagger-sm2 wurde die Fehlerrate für die Fehlergruppe der finiten/infiniten Verben, für die bereits Evaluationsergebnisse vorliegen, um fast ein Drittel gesenkt und für den tnt-

	unverbessert		verbessert	
Tag	Fehlerzahl	Fehlerquote	Fehlerzahl	Fehlerquote
VVFIN	363	4,84%	165	2,20%
VVPP	82	2,74%	70	2,34%
VVINF	95	3,71%	65	2,54%
VAFIN	75	1,58%	24	0,50%
VAINF	26	5,01%	11	2,12%
VMFIN	39	2,40%	14	0,86%
VMINF	14	15,28%	3	3,26%
Summe Fehler Verben	694	3,46%	352	1,76%
Summe Fehler Gesamt	3441	1,95%	3089	1,75%

Tabelle 22.2.: Tagging-Fehler tnt-Tagger-sm1

Tagger-sm1 sogar um fast die Hälfte (vgl. Tab. 22.1 und 22.2, vorletzte Zeile).[7] Damit war das System erfolgreich und hat die Fehlerrate deutlich reduziert. Hervorzuheben ist bei dem Ansatz weiterhin, dass er sich mühelos in das System des flachen Parsens integrieren lässt, da er auf bereits im System verfügbare linguistische Wissensbasen zurückgreift. Zudem nimmt die Tag-Korrektur nur 8,3% der für den Vorgang des flachen Parsings nötigen Laufzeit in Anspruch. Auffällig ist, dass die Verbesserungsquote des Systems mit der Güte des Taggers steigt.[8] Das System kann somit nicht einen gut trainierten Tagger ersetzen, sondern profitiert überdurchschnittlich von besseren Eingabedaten.

[7] Tags, die durch das System verschlechtert wurden, sind in dieser Rechnung bereits eingeschlossen.

[8] Das Sprachmodell des tnt-Tagger-sm2 besteht aus bislang nur einmal manuell korrigierten Daten verschiedener Textsorten.

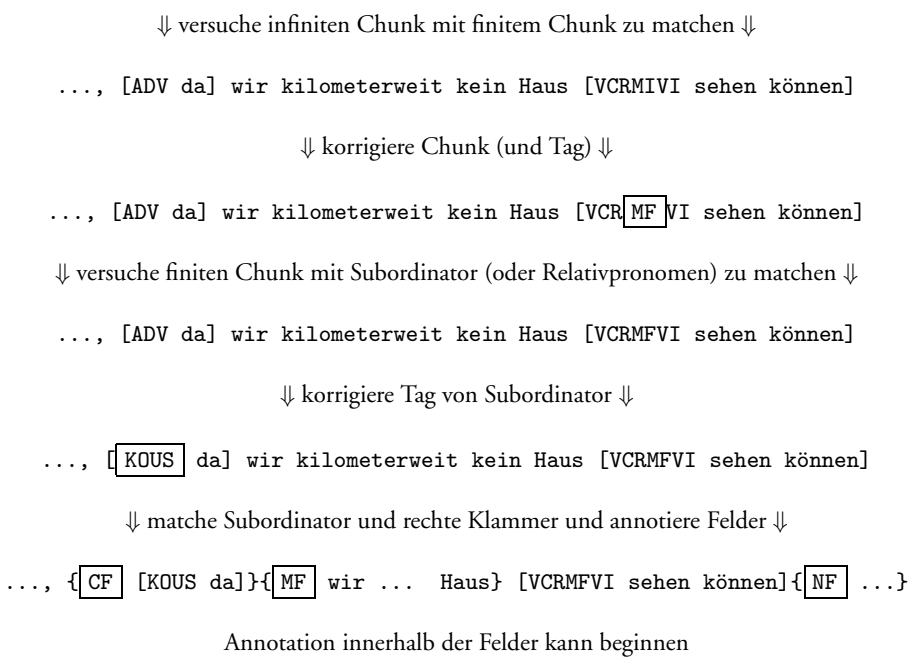

Abbildung 22.3.: Schritte bei der Satzklammererkennung und Tagkorrektur

22.3. Schlussfolgerungen und Ausblick

Die vorliegende Untersuchung hat ein System vorgestellt, das in der Lage ist, durch die Kombination eines statistischen Taggers und eines regelbasierten flachen Parsers beim flachen Parsen deutscher Sätze die Tagging-Fehler deutlich zu reduzieren und so eine verbesserte Grundlage für weitere, tiefere Annotierung zu schaffen. In den nun durch Feld- und Satzgrenzen eingeschränkten Suchräumen wird es einfacher möglich sein, Argumentstrukturen zu erkennen. Das Tag-Korrektur-Verfahren soll noch auf weitere Satzklammerelemente ausgedehnt werden.

Literaturverzeichnis

Abney, S. (1996): "Partial Parsing via Finite-State Cascades". In: *Proceedings of the ESSLLI '96 Robust Parsing Workshop*.

Brants, T. (1998): "TnT – A Statistical Part-of-Speech Tagger". Universität des Saarlandes, Computational Linguistics, Saarbrücken. Online verfügbar: `http://www.coli.uni-sb.de/~thorsten/tnt/`.

Braun, C. (1999): *Flaches und robustes Parsen deutscher Satzgefüge*. Diplomarbeit, Universität des Saarlandes, Saarbrücken.

Bray, T.; Paoli, J.; Sperberg-McQueen, C. M. und Maler, E. (2000): "Extensible Markup Language (XML) 1.0 (Second Edition)". Technischer Bericht, W3C.

Bussmann, H. (1990): *Lexikon der Sprachwissenschaft*. Stuttgart: Kröner, 2. Auflage.

Griesbach, H. (1986): *Neue deutsche Grammatik*. Berlin: Langenscheidt.

Grover, C.; Matheson, C. und Mikheev, A. (1999): "TTT: Text Tokenisation Tool". Language Technology Group, University of Edinburgh, Edinburgh. Online verfügbar: `http://www.ltg.ed.ac.uk/software/ttt/tttdoc.html`.

Helbig, G. und Buscha, J. (1996): *Deutsche Grammatik. Ein Handbuch für den Ausländerunterricht*. Leipzig: Langenscheidt, 17. Auflage.

Hinrichs, E. W.; Kübler, S.; Kordoni, V. und Müller, F. H. (2000): "Robust Chunk Parsing for Spontaneous Speech". In: *Verbmobil: Foundations of Speech-to-Speech Translation*, herausgegeben von Wahlster, W., Berlin: Springer, S. 163–182.

Hirakawa, H.; Ono, K. und Yoshimura, Y. (2000): "Automatic Refinement of a POS Tagger Using a Reliable Parser and Plain Text Corpora". In: *Proceedings of the 18th International Conference on Computational Linguistics, COLING 2000*. International Committee on Computational Linguistics ICCL, Saarbrücken.

Ide, N. (2000): "The XML Framework and Its Implications for Corpus Access and Use". In: *Proceedings of the EAGLES/ISLE Workshop on Meta-Descriptions and Annotation Schemas for Multimodal/Multimedia Language Resources and Data Architectures and Software Support for Large Corpora*. European Language Resources Association, Paris.

Neumann, G.; Braun, C. und Piskorski, J. (2000): "A Divide-and-Conquer Strategy for Shallow Parsing of German Free Texts". In: *Proceedings of ANLP-2000*. Seattle, Washington, S. 239–246.

Peh, L. und Ting, C. H. (1996): "A Divide-and-Conquer Strategy for Parsing". In: *Proceedings of the ACL/SIGPARSE Fifth International Workshop on Parsing Technologies*. S. 57–66.

Rösner, D. (2000): "Combining Robust Parsing and Lexical Acquisition in the XDOC System". In: *KONVENS 2000 Sprachkommunikation*, herausgegeben von Zühlke, W. und Schukat-Talamazzini, E. G. Berlin: VDE Verlag, S. 75–80. ITG-Fachbericht 161.

Schiller, A.; Teufel, S.; Thielen, C. und Stöckert, C. (1995): "Guidelines für das Taggen deutscher Textcorpora mit STTS". IMS Stuttgart und SfS Tübingen, Stuttgart und Tübingen.

Skut, W.; Brants, T.; Krenn, B. und Uszkoreit, H. (1998): "A Linguistically Interpreted Corpus of German Newspaper Texts". In: *ESSLLI Workshop on Recent Advances in Corpus Annotation*. Saarbrücken.

Wauschkuhn, O. (1996): "Ein Werkzeug zur partiellen syntaktischen Analyse deutscher Textkorpora". In: *Natural Language Processing and Speech Technology: Results of the 3rd KONVENS Conference*, herausgegeben von Gibbon, D. Berlin: Mouton de Gruyter, S. 356–368.

Weinrich, H. (1993): *Textgrammatik der deutschen Sprache*. Mannheim: Dudenverlag.