Kontextbasierte lexikalische Kontrolle von Anforderungsdokumenten

Jennifer Krisch
Daimler AG
jennifer.krisch@daimler.com

Abstract

Das Verwalten und Prüfen von Anforderungsdokumenten ist ein sehr wichtiger Bereich in den frühen Phasen von Industrieprojekten. Je früher unpräzise und unvollständige Anforderungen identifiziert werden, desto geringer sind die Folgekosten und die Überarbeitungszeit. Unpräzise und unvollständige Anforderungen resultieren häufig aus der Verwendung von Weak-Words, d.h. von unscharfen Wörtern oder Konstruktionen, die in bestimmten Kontexten Mehrdeutigkeiten auslösen. Um Fehlinterpretationen aufgrund von Weak-Words entgegenzuwirken, müssen diese Wörter oder Konstruktionen identifiziert werden. Eine reine Wortsuche reicht hierbei nicht aus, weil Weak-Words nur in bestimmten Satzkontexten Mehrdeutigkeiten auslösen. Der Beitrag beschreibt eine als Prototyp entwickelte automatisierte Weak-Word-Analysemethodik, durch welche ausgewählte Weak-Words identifiziert werden und bei der aufgrund des Satzkontexts entschieden wird, ob eine Warnung bezüglich eines Mangels an Präzision und Eindeutigkeit des Texts an den Autor zurückgegeben werden soll. Bei der Weak-Word-Analyse kommt ein Lexikon zum Einsatz, in welchem die Weak-Words selbst und deren Kontexte abgelegt sind. Die Evaluation des entwickelten Werkzeugs hat ergeben, dass die Anzahl der Warnungen, die an den Benutzer zurückgegeben werden, durch eine Kontextanalyse im Verhältnis zur wortbasierten Suche erheblich reduziert werden kann, ohne dass problematische Kontexte übersehen werden. Dies stellt eine Verbesserung gegenüber den bisher bekannten Systemen dar.

Keywords: Weak-Words; Kontext-Analyse; Linguistische Annotation

1 Einführung

Das Verfassen von Anforderungen ist in der Industrie der erste Schritt, um von einer Idee zu einem Produkt zu kommen. Die Ideen der Projektbeteiligten müssen angemessen dokumentiert werden, damit jeder weiß was im Projekt gefordert wird. Anforderungstexte sind also „Aussagen über Eigenschaften oder Leistungen eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen“ (Rupp 2007).

Die Qualität einer Anforderung wird durch *Qualitätskriterien* festgelegt. Je mehr Qualitätskriterien von einer Anforderung erfüllt werden, desto höher ist deren Qualität. Anforderungen werden meist in

natürlicher Sprache verfasst, die aber das Risiko von Mehrdeutigkeit, mangelnder Präzision oder Unklarheiten über die rechtliche Verbindlichkeit birgt; daher ist es wichtig, Anforderungen vor allem auf die Qualitätskriterien *Präzision* und *Eindeutigkeit* zu prüfen. Eine Anforderung erfüllt das Qualitätskriterium *Präzision*, „wenn diese keine ungenauen Angaben enthält, wenn wo immer möglich und sinnvoll quantitative Angaben gemacht werden und wenn alle Angaben genauso präzise sind, wie es für die Problemstellung erforderlich ist" (Daimler AG 2013). Das Qualitätskriterium *Präzision* hängt eng mit dem Qualitätskriterium *Eindeutigkeit* zusammen:

> Eine eindeutige Anforderung [sollte] nur auf eine Art und Weise verstanden werden können [...]. Es darf nicht möglich sein, andere Sachverhalte hineinzuinterpretieren. Alle Leser einer Anforderung sollten zu einer einzigen, konsequenten Interpretation der Anforderung gelangen. (Rupp 2007)

Eines der zentralen Qualitätsprobleme von Präzision und Eindeutigkeit ist (fach-) lexikographischer Natur: *Weak-Words*, auch *unscharfe Wörter* genannt, verstoßen in gewissen (Satz-)Kontexten gegen die aufgeführten Qualitätskriterien. Weak-Words sind „Wörter oder Phrasen, deren Benutzung in einem Freitext darauf schließen lässt, dass der Freitext mit hoher Wahrscheinlichkeit unpräzise ist" (Melchisedech 2000). Ein Beispiel für ein Weak-Word ist das Adverb *lang* in der Anforderung, die in Beispiel (1) dargestellt ist.

(1) *Die Taste muss lang gedrückt werden.*

Das Wort *lang* eröffnet hier einen großen Interpretationsspielraum. Ein Projektbeteiligter wird das Wort *lang* als eine Zeitspanne von drei Sekunden interpretieren, ein anderer möglicherweise als eine Zeitspanne von 500 Millisekunden. Dieser Interpretationsspielraum kann im Produktentwicklungsprozess zu großen Problemen führen, weil ein Fehler in der Anforderungsformulierung sich durch viele Teile eines Projekts ziehen kann und möglicherweise erst zu einem sehr späten Zeitpunkt aufgedeckt wird. Aus diesem Grund ist es wichtig, Anforderungen auf das Vorkommen von Weak-Words zu prüfen. Praktisches Ziel der hier beschriebenen Arbeiten ist es, einen Teil dieser Prüfung zu automatisieren.

Die Annahme liegt nahe, dass jedes Auftreten eines Weak-Words die Qualität einer Anforderung mindert, man deshalb Weak-Words identifizieren und Anforderungsautoren bitten sollte, ihre Formulierungen nochmals zu überarbeiten. Es ist aber zu beachten, dass Weak-Words nicht immer die Qualität einer Anforderung mindern. Dies ist im hohen Maße vom Kontext abhängig, wie man in Beispiel (2) sehen kann.

(2) *Die Taste muss 2 Sekunden lang gedrückt werden.*

Hier eröffnet das Weak-Word *lang* keinen Interpretationsspielraum, sondern trägt ganz im Gegenteil dazu bei, die Anforderung zu präzisieren. Bei einer Weak-Word-Analyse muss folglich der Kontext der Weak-Words miteinbezogen werden, um entscheiden zu können, ob diese in einer Anforderung Interpretationsspielraum eröffnen oder nicht (Krisch 2013).

2 Stand der Technik

Es existieren bereits Prüfwerkzeuge, die eine automatisierte Weak-Word-Analyse durchführen. Das sind u.a. *ReQualize* (Heidenreich 2010), *DESIRe*® (Stöckel et al. 2009) und *QuARS* (Lami 2005). Bei allen drei Werkzeugen kann gezielt nach Wörtern gesucht werden, welche in einer Liste abgelegt sind. Wird ein solches Wort in einem Lastenheft gefunden, werden eine Warnung und ein Hinweis an den Benutzer zurückgegeben, dass er die entsprechende Anforderung nochmals überarbeiten sollte. Die Kontexte der Wörter werden bei diesen Werkzeugen nicht in die Analyse miteinbezogen. Bei einer großen Textmenge hat dies die Konsequenz, dass sehr viele Fehlermeldungen produziert werden und der Benutzer sehr oft unnötig gewarnt wird; das führt wiederum dazu, dass das System vom Benutzer nicht akzeptiert wird.
Diese unnötigen Warnungen sollen durch den hier beschriebenen Ansatz zur Erstellung kontextabhängiger Prüf-Regeln für bestimmte Weak-Words reduziert werden.

3 Kontextanalyse und Entwicklung eines lexikalischen Prüfwerkzeugs

3.1 Korpuserstellung

Um die Kontexte ausgewählter Weak-Words identifizieren zu können, musste zunächst ausreichend Textmaterial bereitgestellt werden. Hierfür wurden Lastenhefte aus einer Datenbank der potentiellen Anwender exportiert, durch ein eigens entwickeltes Werkzeug tokenisiert (Identifikation von Wort- und Satzgrenzen) und linguistisch annotiert (mit den *mate*-Werkzeugen, (Bohnet 2009)). Die tokenisierten Lastenhefte wurden zunächst lemmatisiert, anschließend mit Wortart- und Morphologie-Annotationen versehen und zuletzt noch auf der Grundlage von Dependenzsyntax analysiert (geparst). Abbildung 1 zeigt ein Beispiel eines Anforderungssatzes nach der computerlinguistischen Annotation. In schwarz ist die jeweilige Satz- bzw. Wortnummer dargestellt und in dunkelblau die Wortform (Token). Orange markiert sind die Lemmata, grün die Wortart-Tags, pink die Morphologie-Tags, rot die Abhängigkeiten zwischen den einzelnen Tokens und hellblau die syntaktischen Informationen. Das Wort *Wiedereinschalten* beispielsweise ist das zweite Token im ersten Satz mit der ID *SPM-8014*. Die Grundform, also das Lemma, ist *Wiedereinschalten*. Außerdem ist es ein Nomen, hat den Kasus Nominativ, steht im Singular und ist neutrum. *Wiedereinschalten* bezieht sich auf Token 3, also auf *erfolgt*, und ist dessen Subjekt.
Die linguistische Annotation der Anforderungen dient vor allem zur Generalisierung der in Abschnitt 3.2 entwickelten Regeln.

```
1_1_SPM-8014    Das     der     ART     case=nom|number=sg|gender=neut  2       NK
1_2_SPM-8014    Wiedereinschalten       Wiedereinschalten       NN      case=nom|num-
 ber=sg|gender=neut 3       SB
1_3_SPM-8014    erfolgt erfolgen                VVFIN   number=sg|person=3|tense=pres|-
 mood=ind   0       --
1_4_SPM-8014    immer   immer   ADV     _       3       MO
1_5_SPM-8014    durch   durch   APPR    _       3       MO
1_6_SPM-8014    eine    ein     ART     case=acc|number=sg|gender=fem           8
 NK
1_7_SPM-8014    erneute         erneut  ADJA    case=acc|number=sg|gender=fem|de-
 gree=pos   8       NK
1_8_SPM-8014    Anforderung     Anforderung     NN      case=acc|number=sg|gender=fem
      5             NK
1_9_SPM-8014    vom     von     APPRART         case=dat|number=sg|gender=neut  8
 MNR
1_10_SPM-8014   Logikträger    Logikträger     NN      case=dat|number=sg|gender=neut
 9          NK
1_11_SPM-8014   .       --      $.      _       10      --
```

Abbildung 1: Darstellung eines Anforderungs-Satzes mit linguistischen Annotationen.

Insgesamt wurde mit diesem Vorgehen ein Korpus erzeugt, das 88 166 Sätze und 990 011 Tokens enthält. Auf Grundlage dieses Korpus wurden Experimente zur Identifikation präziser und unpräziser Kontextmuster ausgewählter Weak-Words durchgeführt. Für die Evaluation wurde ein Korpus mit neuen Lastenheften erzeugt (vgl. Kapitel 4).

3.2 Ermittlung der Kontexte ausgewählter Weak-Words

Die verwendete Arbeitsmethodik ist von der korpusbasierten Lexikographie inspiriert: Zur Entwicklung lexemspezifischer Regeln für die Identifikation von Weak-Words im Kontext wurde das Anforderungs-Korpus interaktiv mit dem Suchwerkzeug CQP (Evert & Hardie 2011) durchsucht. In Tabelle 1 sind Beispiele für „gute“ und „problematische“ Kontexte des potentiellen Weak-Words *mal* dargestellt.

Zusammensetzung	Beispiel
CARD mal	3 mal
erst.+ mal	das erste Mal
XXX_X... mal	<Parameter> mal
tbdX (default CARD) mal	tbd1 (default 10) mal
[0-9]+ten mal	nach dem 3ten Mal
nächste.* mal	das nächste Mal
letzt.+ mal	beim letzten Mal
[a-z] mal	x mal
jedes mal wenn	jedes mal wenn
auch mal	auch mal

Tabelle 1: Kontextmuster des Weak-Words *mal/Mal.*

Die grün markierten Spalten sind Kontextmuster von *mal*, in welchen die Verwendung des Wortes nicht dazu führt, dass die Anforderung unpräzise wird. Angaben wie *3 mal* oder *nach dem 3ten Mal* sind präzise und schaden der Qualität einer Anforderung nicht. Auch Kontextmuster, bei denen ein Parameter vor dem Wort *mal* steht, sind in Ordnung. Genauso verhält es sich mit Konstruktionen, die eine flektierte Form von *erst* und ein nachfolgendes *Mal* enthalten und mit Konstruktionen, die die Form *tbdX (default 3) mal*[1] aufweisen. Die rot markierten Muster hingegen sollten in Anforderungen vermieden werden, da sie die Verständlichkeit der betreffenden Anforderungen mindern. Ein Beispiel hierfür ist in (3) gegeben.

(3) *Die Taste kann auch mal klemmen.*

In (3) wird nicht eindeutig spezifiziert, wann dieser Fall, dass die Taste klemmen kann, genau auftreten kann. Das Weak-Word *mal* eröffnet einen großen Interpretationsspielraum und aus diesem Grund müssen Anforderungen wie diese zur nochmaligen Überarbeitung an den Benutzer zurückgeliefert werden.

Die orange markierten Muster sind Fälle, bei denen aufgrund des Satzkontexts alleine nicht sicher ist, ob sie präzise genug sind. In manchen Fällen wird es klar sein, wann *das nächste Mal* oder *das letzte Mal* ist, in anderen Fällen wiederum nicht. Hier sollte zur Sicherheit eine Warnung an den Benutzer ausgegeben werden.

Der Grundgedanke der Weak-Word-Analyse ist, dass nur die wirklich präzisen Konstruktionen mit potentiellen Weak-Words dem Benutzer nicht zurückgeliefert werden sollen. Alle anderen Konstruktionen (rot und orange markiert) sollten an den Benutzer zurückgegeben werden, damit dieser sich die betreffende Anforderung nochmals anschauen und sie überarbeiten kann. Dies kann zwar dazu

1 Konstruktionen wie diese sind Standardkonstruktionen in Lastenheften und müssen deshalb nicht als Warnung an den Benutzer zurückgegeben werden.

führen, dass weiterhin unnötige Warnungen an den Benutzer geliefert werden, dafür wird aber auch keine unpräzise Anforderung übergangen.

In gleicher Weise wurden auch die Kontextmuster für das Weak-Word *lang* identifiziert. In Tabelle 2 sind ausschließlich die präzisen Kontextmuster von *lang* angegeben. Insgesamt wurden acht unpräzise Kontextmuster und drei Kontextmuster ermittelt, bei denen nicht klar entschieden werden konnte, ob sie immer zu präzisen Anforderungen führen oder nicht. Konstruktionen wie *länger als 3 Sekunden* und *langes Drücken von 1 Sekunde* sind präzise und lösen keinen Interpretationsspielraum aus. Ein ebenfalls präzises Beispiel ist *länger als für die Nachlaufzeit*. Hier weist der bestimmte Artikel darauf hin, dass der Begriff *Nachlaufzeit* in einer vorangehenden Anforderung schon verwendet wurde und somit spezifiziert sein sollte. An dieser Stelle wird zwar keine Warnung an den Autor ausgegeben, am Schluss einer Analyse werden aber dem Autor alle Nomina zurückgegeben, denen ein bestimmter Artikel vorangeht und die mit dem Wort *lang* zusammenhängen, mit der Bitte um Prüfung, ob die Wörter eindeutig beschrieben sind. Tritt hingegen eine Konstruktion mit unbestimmtem Artikel auf wie in *länger als eine Erholungszeit*, wird in jedem Fall eine Warnung ausgegeben, da der unbestimmte Artikel darauf hinweist, dass das Wort *Erholungszeit* an dieser Stelle neu eingeführt wird und noch nicht vollständig beschrieben ist.

Für die Weak-Words *kurz* und *schnell* wurden dieselben Regeln verwendet, die auch bei *lang* zum Tragen kommen. Ziel war es zu testen, ob sich Regeln eines Weak-Words auf semantisch ähnliche Weak-Words übertragen lassen, d.h. ob die definierten Regeln generalisierbar sind. Die Weak-Words und die korrespondierenden Regeln wurden in einem Lexikon abgelegt, in einen ersten Prototypen implementiert und anschließend evaluiert.

Zusammensetzung	**Beispiel**
lang als ca.	länger als ca. 3 Sekunden
CARD NN/NE lang	15 Sekunden lang
lang als CARD	länger als 3 Sekunden
lang als für def.ART	länger als für die Nachlaufzeit
wie lang	wie lange
lang als def.ART	länger als die Verzögerungszeit
XXX_X_... lang	<Parameter> lang
lang VERB als	nicht länger betrieben als
lang als XXX_X_...	länger als <Parameter>
lang CARD NN/NE	länger 500 Millisekunden
lang NN/NE (</> ...)	langes Drücken (> 1 Sekunde)
lang NN/NE von	langes Drücken von 1 Sekunde
lang def.ART NN/NE	länger der Nachlaufzeit

Tabelle 2: Präzise Kontextmuster des Weak-Words *lang*.

Für jedes Weak-Word wurde ein Lexikoneintrag mit den entsprechenden Regeln erstellt. Die Kontextregeln sind als Suchmuster in der Skriptsprache Python, auf der Basis der in Abschnitt 3.1 illustrierten Annotationen, implementiert. Das Lexikon enthält auf dem Stand von Frühjahr 2014 die Weak-Words *mal*, *lang*, *kurz* und *schnell*. Neue Wörter und zugehörige Regeln können analog hinzugefügt werden. Im Lexikon stehen nur die präzisen Kontextmuster der Weak-Words. Dies gewährleistet, dass alle präzisen Kontexte vom System erkannt werden und alle anderen Kontexte, die nicht im Lexikon vermerkt sind, an den Benutzer zurückgegeben werden. Dadurch werden einerseits Fehlermeldungen, die bei anderen Tools unnötigerweise ausgegeben werden, vermieden, und andererseits keine unpräzisen Anforderungen übergangen.

4 Evaluation

Die Regeln wurden anhand eines Korpus von insgesamt 99 955 Sätzen und 1 160 409 Tokens evaluiert. In diesem Korpus kam das Weak-Word *mal* 44 mal vor, *lang* trat 147 mal, *kurz* 106 mal und *schnell* 207 mal auf. Es wurden alle Wortformen der vier Weak-Words analysiert, nicht nur die Grundformen.
Tabelle 3 zeigt die Auswertung der Ergebnisse für die 44 Erscheinungsstellen von *mal*. 26 Anforderungen wurden automatisch korrekt als präzise Anforderung klassifiziert. 15 unpräzise Anforderungen wurden ebenfalls richtig als unpräzise Anforderungen klassifiziert. Drei präzise Anforderungen hingegen wurden fälschlicherweise als unpräzise interpretiert, d.h. es wurden insgesamt drei überflüssige Warnungen an den Benutzer zurückgegeben. Interessant ist vor allem, dass keine unpräzise Anforderung übersehen wurde.
Das Ziel der Weak-Word-Analyse ist es, die unnötigen Warnungen an den Benutzer zu reduzieren, es soll dabei aber keine unpräzise Anforderung übergangen werden. Durch die Formulierung von Kontextregeln des Weak-Words *mal* für unproblematische Fälle konnten gegenüber einer wortbasierten Strategie 26 Warnungen entfallen, was fast 60 % der Gesamtmenge an Warnungen des wortbasierten Systems entspricht.

automatische Analyse →

manuelle Analyse ↓

	Unpräzise Anforderung	**Präzise Anforderung**
Unpräzise Anforderung	15	0
Präzise Anforderung	3	26

Tabelle 3: Evaluation der Ergebnisse für das Weak-Word *mal*.

Ein Beispiel für eine falsch klassifizierte Anforderung, die das Wort *mal* enthält, ist in (4) gegeben.

(4) *Die Funktion darf nicht ein viertes Mal gestartet werden.*

Die beiden anderen falsch klassifizierten Anforderungen sind von derselben Art wie (4). Der POS-Tagger interpretiert das Zahlwort *viertes* nicht als Zahl (Tag = *CARD*), sondern als Artikel. Eine Konstruktion wie diese wird bisher nicht mit den Regeln aus Tabelle 1 abgefangen. Eine entsprechende Regel sollte in zukünftigen Arbeiten entworfen und in das Lexikon integriert werden.
In Tabelle 4 sind die entsprechenden Precision- und Recall-Werte und die daraus resultierenden F1-Measure-Werte für die Klassen *Unpräzise Anforderung* und *Präzise Anforderung* zu sehen. Der Precision-Wert gibt den Anteil der Suchergebnisse an, die tatsächlich relevant sind in Bezug auf die Gesamtgröße des Ergebnisses (Dörre, Gerstl, & Seiffert 2004). Der Precision-Wert der Klasse *Unpräzise Anforderung* liegt beispielsweise bei 83,33 %. Der Grund hierfür ist, dass zwar alle relevanten Anforderungen gefunden wurden, aber noch überflüssige Warnungen an den Benutzer geliefert werden. Dies ist bei der Klasse *Präzise Anforderung* nicht der Fall, und deshalb beträgt hier der Precision-Wert 100 %.
Der Recall-Wert bezeichnet den Anteil der relevanten Suchergebnisse in Bezug auf die Menge aller für diese Suche relevanten Ergebnisse (Dörre, Gerstl, & Seiffert 2004). Für die Klasse *Präzise Anforderung* werden nicht alle relevanten Instanzen gefunden, für die Klasse *Unpräzise Anforderung* allerdings schon, was die Werte von 89,66 % und 100 % erklärt.

	Klasse: Präzise Anforderung	Klasse: Unpräzise Anforderung
Precision	100 %	83,33 %
Recall	89,66 %	100 %
F_1-Measure	94,55 %	90,91 %

Tabelle 4: Precision, Recall und F_1-Measure: *mal*.

Das F1-Maß ist eine Kombination aus Precision und Recall (geometrisches Mittel) und gibt einen Gesamtüberblick über die Performanz des entwickelten Werkzeugs in Bezug auf die einzelnen Klassen.
Tabelle 5 zeigt die Auswertung der Ergebnisse für das Weak-Word *lang*. 129 Kontexte von *lang* konnten korrekt klassifiziert werden. Es gab nur 18 Fehlklassifizierungen. Insgesamt konnten 69 unnötige Warnungen vermieden werden, was in der Gesamtmenge knapp 50 % entspricht.

automatische Analyse →

manuelle Analyse ↓

	Unpräzise Anforderung	**Präzise Anforderung**
Unpräzise Anforderung	60	0
Präzise Anforderung	18	69

Tabelle 5: Evaluation der Ergebnisse für das Weak-Word *lang*.

Die Fehlanalysen beruhen vor allem auf Klammer- und Adjunkt-Konstruktionen, die vom System bisher nicht behandelt werden können. Ein Beispiel für eine Klammerkonstruktion, die falsch klassifiziert wurde, ist in (5) zu sehen.
(5) *Die Eingriffe sollen nicht innerhalb einer applizierbaren langen Vergangenheit (5 Sekunden) liegen.*

Für Konstruktionen wie diese wurden bisher keine Regeln formuliert. Inwiefern hier eine Regel entworfen werden kann, soll in weiteren Arbeiten untersucht werden.

Weak-Word	*lang*		*kurz*		*schnell*	
Klasse	*Präz. Anf.*	*Unpräz. Anf.*	*Präz. Anf.*	*Unpräz. Anf.*	*Präz. Anf.*	*Unpräz. Anf.*
Precision	100 %	76,92 %	100 %	82,83 %	100 %	98,47 %
Recall	79,31 %	100 %	29,12 %	100 %	78,57 %	100 %
F_1-Measure	88,46 %	86,95 %	45,11 %	90,61 %	88 %	99,23 %

Tabelle 6: Precision, Recall und F_1-Measure: *lang*, *kurz*, *schnell*.

Eine Adjunkt-Konstruktion, die ebenfalls nicht korrekt klassifiziert werden konnte, ist in Beispiel (6) gegeben.
(6) *War die Strecke bis zu Position A zu lang, d.h. es wurden mehr als drei Hallimpulse gezählt, wird das Glasdach geschlossen.*

Bei Beispiel (6) handelt es sich um eine linguistisch sehr komplexe Anforderung. Regeln für eine Konstruktion wie diese zu definieren, ist maschinell nur schwer umsetzbar. Man sollte sich hier auch die Frage stellen, ob sich der Aufwand eine Regel zu finden lohnt, oder ob es nicht doch sinnvoll ist, den Autor auf eine so komplizierte Formulierung wie in (6) hinzuweisen, damit er die Anforderung nochmals in eine verständlichere Konstruktion umwandelt.
Tabelle 6 zeigt die Auswertung der Ergebnisse der Weak-Words *lang*, *kurz* und *schnell*. Für die Analyse von *kurz* und *schnell* wurden dieselben Regeln verwendet, die auch bei *lang* zum Tragen kommen. Die Ergebnisse von *lang* und *schnell* sind solide. Bei *kurz* kann man einen deutlichen Einbruch erkennen. Zwar generalisieren die Kontexttypen von *lang* gut auf *schnell* und ordentlich auf *kurz*, aber die drei Wörter werden durchaus unterschiedlich verwendet. Regeln, die für jedes einzelne Wort separat festgelegt werden, könnten also noch bessere Ergebnisse liefern. Auch hier resultieren die Fehlanalysen des Systems vor allem aus Klammer- und Adjunkt-Konstruktionen.

5 Zusammenfassung und Ausblick

Es wurde ein sehr stark auf Kontext-Typen aus Anforderungstexten zugeschnittenes Lexikon von potentiell unspezifischen Wörtern erstellt, welches automatisch anwendbar ist. Das Lexikon enthält Wörter und deren Kontextmuster; die Arbeitsmethodik ist von der korpusbasierten Lexikographie inspiriert, zielt aber auf sehr präzise, automatisch überprüfbare Muster von Kontexttypen. Die entwickelte Weak-Word-Analyse, die dieses Lexikon als Quelle heranzieht, liefert solide und über semantisch verwandte Wörter generalisierbare Ergebnisse.

In weiteren Arbeiten sollen weitere Weak-Words in die Analyse integriert und neue Lexikoneinträge mit zugehörigen Regeln erzeugt werden. Für zukünftige Arbeiten wäre es auch interessant zu untersuchen, welche anderen linguistischen Probleme, neben den Weak-Words, gegen die Qualitätskriterien verstoßen und ob auch diese automatisiert behandelt werden können.

6 References

Bohnet, B. (2009). Efficient Parsing of Syntactic and Semantic Dependency Structures. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning, CoNLL 2009,* Boulder, Colorado, US.

Daimler AG (2013). Qualitätskriterien, Daimler AG, Böblingen, DE.

Dörre, J., Gerstl, P., & Seiffert, R. (2004). Volltextsuche und Text Mining. In K.-U. Carstensen, C. Ebert, C. Endriss, S. Jekat, R. Klabunde, & H. Langer, Computerlinguistik und Sprachtechnologie: Eine Einführung. München: Spektrum, pp. 479-495.

Evert, S., Hardie, A. (2011). Twenty-first century Corpus Workbench: Updating a query architecture for the new millennium. In *Proceedings of the Corpus Linguistics 2011 conference*. University of Birmingham, UK.

Heidenreich, M. (2010). Metriken und Werkzeugunterstützung zur Überprüfung von Anforderungen. OBJEKTspektrum, Ausgabe RE/2010.

Krisch, J. (2013). Identifikation kritischer Weak-Words aufgrund ihres Satzkontextes in Anforderungsdokumenten. Diplomarbeit. Institut für maschinelle Sprachverarbeitung, Universität Stuttgart, Stuttgart, DE.

Lami, G. (2005). QuARS: A Tool for Analyzing Requirements. Carnegie-Mellon Software Engineering Institute.

Melchisedech, R. (2000). Verwaltung und Prüfung natürlichsprachlicher Spezifikationen. Dissertation. Universität Stuttgart, Stuttgart, DE.

Rupp, C. (2007). Requirements-Engineering und -Management: professionelle, iterative Anforderungsanalyse für die Praxis. München: Hanser Verlag.

Stöckel, F., Stolz, P., Uddin, I., & Endriss, L. (2009). DESIRe® Dynamic Expert System for Improving Requirements. HOOD Group.