KLASSIFIKATION VON ANFORDERUNGEN UND INFORMATIONEN ZUR UNTERSTÜTZUNG VON QUALITÄTSSICHERUNGSPROZESSEN

INAUGURAL–DISSERTATION

zur

Erlangung des Doktorgrades

der Mathematisch-Naturwissenschaftlichen Fakultät

der Universität zu Köln

vorgelegt von

Jonas Paul Winkler

aus Potsdam

Köln

2021

Berichterstatter (Gutachter):
Prof. Dr. Andreas Vogelsang
Prof. Dr. rer. nat. Kurt Schneider

Tag der mündlichen Prüfung: 23.04.2021

ZUSAMMENFASSUNG

Anforderungsdokumente werden im Anforderungsmanagement verwendet, um Eigenschaften und Verhalten von Systemen zu dokumentieren. In der Automobilindustrie werden diese Dokumente verwendet, um die von Zulieferern zu fertigenden Komponenten zu beschreiben und um für die Kommunikation zwischen Zulieferer und Konzern eine rechtliche Grundlage zu schaffen. Daher müssen diese Dokumente diversen Qualitätsstandards und Qualitätsrichtlinien entsprechen.

In manuellen Reviews werden Anforderungsdokumente gegen diese Richtlinien geprüft. Eine Richtlinie besagt, dass in Anforderungsdokumenten eine klare Trennung zwischen rechtlich verbindlichen Anforderungen und sogenannten Zusatzinformationen (Abbildungen, Erläuterungen, Beispiele, Verweise, etc.) existieren muss. Dazu wird jedes Objekt entsprechend dem Inhalt mit einem *Objekttyp* annotiert. Die Überprüfung der Korrektheit des Objekttyps ist ein zeitaufwändiger und fehleranfälliger Prozess, da Anforderungsdokumente in der Regel mehrere tausend Objekte umfassen.

In dieser Arbeit wird am Beispiel des Reviews des Objekttyps untersucht, ob und in welcher Art und Weise der Reviewprozess durch den Einsatz von maschinellem Lernen unterstützt werden kann. Dazu wird zuerst ein Klassifikator trainiert, der in der Lage ist, zwischen Anforderungen und Zusatzinformationen zu unterscheiden. Ein darauf basierendes Werkzeug ist in der Lage, Anwender bei der Überprüfung des Objekttyps durch Hinweise und Warnungen zu unterstützten.

In empirischen Studien wird untersucht, ob Anwender durch den Einsatz des Werkzeugs das Review von Anforderungsdokumenten besser durchführen können. Die Ergebnisse zeigen, dass Anwender nicht nur mehr falsch klassifizierte Objekte finden, sondern auch durchschnittlich 60% der für das Review verwendeten Zeit einsparen können.

Durch die Übertragung des Ansatzes auf ein weiteres Klassifikationsproblem wird zudem gezeigt, dass der Einsatz von Werkzeugen nicht nur auf den Anwendungsfall Objekttypklassifikation beschränkt ist, sondern potenziell auf viele weitere zu überprüfende Richtlinien übertragbar ist.

ABSTRACT

Requirements specifications are used in requirements management to document properties and behaviour of systems. In the automotive domain, these documents are used to describe the components that suppliers must manufacture and define the rights and liabilities of both the supplier and automotive company. Therefore, these documents must adhere to various quality standards and guidelines.

Requirement specifications are verified against these quality standards in manual reviews. One specific guideline requires that within a requirements specification, requirements must be clearly separated from other content ("additional information") such as figures, explanations, examples and references. To achieve this, each object in a specification is marked with an object type attribute. The review of this attribute is a time intensive and error prone process since requirements specifications usually contain more than 1.000 objects each.

In this thesis, we examine the question if and how machine learning techniques can support requirements engineers during review. First, a classifier is trained on a dataset containing requirements and information objects. This classifier is able to distinguish between these two object types. A tool that incorporates this classifier is constructed with the goal to aid requirements engineers in performing the review by providing hints and warnings.

We analyse the ability of the tool to support requirements engineers by performing several empirical studies. The results imply that requirements engineers may find more wrongly classified objects and save about 60% of the time used for manual reviews.

The approach is applied to another classification task. This shows that the usage of tools to support manual reviews is not limited to the object type classification task but can be adapted to many more guidelines that need to be checked.

PUBLIKATIONEN

Diese Arbeit baut auf den Ergebnissen der folgenden Publikationen auf.

[1] Jonas P. Winkler. „Automatische Klassifikation von Anforderungen zur Unterstützung von Qualitätssicherungsprozessen“. In: *INFORMATIK 2016*. Hrsg. von Heinrich C. Mayr und Martin Pinzger. Bonn: Gesellschaft für Informatik e.V., 2016, S. 1537–1549. DOI: 10.14279/depositonce-6970.

[2] Jonas P. Winkler, Jannis Grönberg und Andreas Vogelsang. „Optimizing for Recall in Automatic Requirements Classification: An Empirical Study“. In: *Proceedings of the 27th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2019, S. 40–50. DOI: 10.1109/RE.2019.00016.

[3] Jonas P. Winkler, Jannis Grönberg und Andreas Vogelsang. „Predicting How to Test Requirements: An Automated Approach“. In: *Proceedings of the 27th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2019, S. 120–130. DOI: 10.1109/RE.2019.00023.

[4] Jonas P. Winkler und Andreas Vogelsang. „Automatic Classification of Requirements Based on Convolutional Neural Networks“. In: *3rd IEEE International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*. 2016, S. 39–45. DOI: 10.1109/REW.2016.021.

[5] Jonas P. Winkler und Andreas Vogelsang. „What Does My Classifier Learn? A Visual Approach to Understanding Natural Language Text Classifiers“. In: *Proceedings of the 22nd International Conference on Natural Language & Information Systems*. NLDB. 2017, S. 468–179. DOI: 10.1007/978-3-319-59569-6_55.

[6] Jonas P. Winkler und Andreas Vogelsang. „Using Tools to Assist Identification of Non-Requirements in Requirements Specifications - A Controlled Experiment“. In: *Requirements Engineering: Foundation for Software Quality: 24th International Working Conference*. 2018. DOI: 10.1007/978-3-319-77243-1_4.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

ABKÜRZUNGEN

ASET	Automated System Engineering Technologies
CNN	Convolutional Neural Network
DCR	Defect Correction Rate
DIM	Document Influence Matrix
DIR	Defect Introduction Rate
DOORS	Dynamic Object Oriented Requirements System
EARS	Easy Approach to Requirements Syntax
GloVE	Global Vectors
HDMR	Hidden Defect Miss Rate
HFA	Hands Free Access
IIM	Intermediate Influence Matrix
IM	Influence Matrix
Inf	Information
KDD	Knowledge Discovery in Databases
KG	Kontrollgruppe
KLH	Komponentenlastenheft
LSI	Latent Semantic Indexing

LSTM	Long Short Term Memory
ML	Machine Learning
NLP	Natural Language Processing
NN	Neural Network
ORR	Object Rephrase Rate
PCA	Principal Component Analysis
PVM	Potential Verification Method
RE	Requirements Engineering
Req	Requirement / Anforderung
SIM	Starting Influence Matrix
SLH	Systemlastenheft
SVM	Support Vector Machine
TPO	Time per Object
WG	Werkzeuggruppe
WL	Window Lift
WWC	Wiper and Washing Controller

1 EINLEITUNG

In den letzten Jahren hat es in der Automobildomäne erstaunliche Fortschritte gegeben. Viele neuartige Systeme, wie beispielsweise Brems-, Abstands- und Spurhalteassistenten, Infotainment-Systeme und Systeme zur Teilautomatisierung bestimmter Fahrszenarien haben das Automobil durch ihren Beitrag zum Fahrkomfort des Fahrers und der Fahrsicherheit revolutioniert [109]. Der Trend geht dabei noch weiter. In Zukunft sollen Autos vollständig automatisiert durch Städte und über Autobahnen fahren und Personen effizient und komfortabel zum Ziel bringen.

Die Vorteile liegen auf der Hand: Durch die hohe Automatisierung können Gefahrensituationen schneller erkannt werden. Eine Studie hat gezeigt, dass durch den Einsatz von Abstandshaltern und Kollisionswarnsystemen in 51% der Fahrzeuge die Anzahl der Gefahrensituationen auf Autobahnen um 32% bis 82% reduziert werden kann [109]. Insbesondere in Kombination mit Car2X-Kommunikationssystemen können Fahrzeuge durch die Kommunikation von Staus, Baustellen und Unfällen effizienter zum Ziel navigiert werden.

All diese Systeme werden durch Software gesteuert. Der Umfang der Software, die dadurch bedingt auf den Steuergeräten des Automobils untergebracht werden muss, hat sich wegen der vielen Softwaresysteme in den letzten Jahren exponentiell entwickelt [22]. Die Rolle der Software hat in unterschiedlichsten Gebieten der Automobilentwicklung viele neue Herausforderungen geschaffen:

> „In a time of only 30 years the amount of software related development activities went from 0 to 30 or even 40%. If we assume that an engineer works about 35 to 40 years in industry, it is obvious that the companies where not able to gather sufficient competencies quickly enough."
>
> – *Manfred Broy, 2006* [22]

Die unterschiedlichen Herausforderungen sind sehr vielseitig:

- Durch den wachsenden Softwareanteil benötigt die Automobilindustrie neue Kompetenzen aus dem Bereich der eingebetteten Softwareentwicklung.
- Die Komplexität der Funktionalität moderner Fahrzeuge erfordert eine geeignete Organisation in Ebenen und eine geeignete Wahl unterschiedlicher Abstraktionslevels.

- Die Hardware- und Kommunikationsinfrastruktur von Fahrzeugen muss angepasst werden, um den wachsenden Bedarf an Kommunikation zwischen den Steuergeräten zu ermöglichen.
- Die Entwicklungsprozesse der Automobilindustrie müssen an die Prozessanforderungen von Softwareentwicklungsprozessen angepasst werden.

Herausforderung Anforderungsmanagement

Eine der größten Prozessherausforderungen ist es allerdings, ein geeignetes Anforderungsmanagement zu finden. Besonders bei softwareintensiven Systemen ist der Spielraum für Lösungen größer als je zuvor [22]. Anforderungsdokumente legen die Eigenschaften einer Lösung fest, entstehen als Ergebnis des Anforderungsmanagements und spielen in der Automobilentwicklung als Kommunikationsmittel zwischen Konzern und Zulieferer eine zentrale Rolle. Die Prozesse im Anforderungsmanagement müssen dabei an die Besonderheiten der Softwareentwicklung angepasst werden, wie zum Beispiel die Tatsache, dass Software sehr schnittstellenintensiv ist und die Entwicklung oftmals an viele verschiedene Drittentwickler ausgelagert wird (Outsourcing).

Qualität von Anforderungsdokumenten

Ebenfalls muss während der Entwicklung darauf geachtet werden, dass die Anforderungsdokumente den üblichen Qualitätsstandards (notwendig, lösungsneutral, eindeutig, konsistent, vollständig, atomar, praktikabel, nachverfolgbar und überprüfbar) entsprechen [52]. Mehrere Studien zeigen, dass Unternehmen trotz der enormen Wichtigkeit von hochqualitativen Anforderungsmanagement Schwierigkeiten damit haben, geeignete Anforderungsmanagementprozesse zu definieren und umzusetzen [38, 39]. Als Ursachen dafür werden unter anderem unzureichende oder mangelhafte Kommunikation zwischen den Stakeholdern, sich während eines Projekts verändernde Projektziele oder auch ein innerhalb eines Unternehmens variierendes Verständnis über die nötige Detailtiefe von Anforderungen genannt.

Folglich ist die Qualität von in der Industrie anzutreffenden Lastenheften nur selten sehr gut. In einer Studie [74] zur Umsetzung funktionaler Sicherheit in insgesamt 15 Unternehmen stellte sich beispielsweise heraus, dass nur selten vollständige Anforderungsdokumente zu Entwicklungsbeginn vorliegen und diese teilweise fehlerhaft sind.

Die Qualität von Anforderungsdokumenten ist wichtig, da Anforderungsdokumente mit niedriger Qualität oder fehlerhaften Inhalten weitere Fehler in auf den Anforderungsdokumenten aufbauenden Dokumenten (z.B. Testspezifikationen) und Prozessen (z.B. Architekturentwurf) provozieren [83]. Werden Fehler in Anforderungsdokumenten erst spät entdeckt, ist es in der Regel sehr kostenintensiv, diese Fehler in den Anforderungsdokumenten und allen darauf aufbauenden Artefakten zu beseitigen. Diese Kosten steigen meist exponentiell. Kostet die Behebung eines Fehlers in der Definitionsphase

der Anforderungen beispielsweise 100 Euro, können es während des Systembetriebs bereits 10.000 Euro sein [88, S. 56].

Qualitätssicherung durch Werkzeuge

Die Qualität von Anforderungen kann durch den gezielten Einsatz von speziellen Anforderungsmanagementwerkzeugen gesteigert werden. In einer Fallstudie [76] wurden dazu Standardwerkzeuge (Office-Software zur Dokumentation, einfache Grafikprogramme zur Erstellung schematischer Darstellungen und E-Mail zur Kommunikation) mit spezialisierten Anforderungsmanagementwerkzeugen (RequisitePro) verglichen. Die mit speziellen RE-Werkzeugen erstellten Dokumente wurden in vielen der definierten Evaluationskriterien besser bewertet. Dies wird durch eine aktuelle Studie bestätigt [40], in der unter anderem die Fähigkeit aktueller RE-Werkzeuge, Anforderungen auf Fehler zu untersuchen, analysiert wurde. Dabei schnitten viele der untersuchten Werkzeuge gut bis sehr gut ab.

Es ist daher kaum verwunderlich, dass intensiv an der Entwicklung von Werkzeugen zur Unterstützung der Prozesse im Anforderungsmanagement geforscht wird. Femmer u. a. [37] stellen beispielsweise ein Werkzeug vor, welches nach sogenannten *Requirements-Smells* in Anforderungsdokumenten sucht. Requirements-Smells sind schlecht gewählte Formulierungen, die häufig in Anforderungen vorkommen und die jeweilige Anforderung unpräzise, mehrdeutig oder nicht testbar machen. Das Werkzeug zeigt diese Fehler dem Benutzer an und motiviert ihn dazu, die Fehler zu beheben. Ahmed und Kanwal [5] vergleichen mehrere Werkzeuge, welche die Gewinnung von Anforderungen durch visuelle Werkzeuge unterstützen. Andere Arbeiten stellen automatische Ansätze zur Suche nach Trace-Links vor [72]. Dazu werden Information-Retrieval-Methoden eingesetzt. In allen Beispielen wurde durch eine Benutzerstudie gezeigt, dass Benutzer die jeweiligen Aufgaben mit den Werkzeugen effektiver und/oder effizienter durchführen konnten.

In dieser Arbeit wird ebenfalls ein Ansatz zur Anforderungsanalyse mit dem Ziel, die Qualität von Anforderungen zu steigern, vorgestellt. Ziel des Ansatzes ist es, eine klare Trennung zwischen Anforderungen und Nicht-Anforderungen zu schaffen und schwach formulierte Anforderungen ausfindig zu machen. Im Kern basiert der Ansatz auf künstlichen neuronalen Netzen, mit denen Anforderungen auf Wortebene analysiert werden.

1.1 ANFORDERUNGSQUALITÄT AN BEISPIELEN

Gemäß ISO-Norm 29148 [52] ist Anforderungsmanagement[1] eine Aktivität zwischen zwei Parteien, mittels der die Anforderungen an ein System, eine Software oder einen Service identifiziert und dokumen-

1 Engl. Requirements Engineering. Anforderungsmanagement wird in dieser Arbeit als übergreifender Begriff für alle Aktivitäten im Requirements Engineering verwendet.

tiert werden. Anforderungen sind Aussagen, die in einem System umgesetzt werden müssen. Ein Anforderungsdokument, bestehend aus einer Sammlung von Anforderungen, legt die Verbindlichkeiten zwischen einem Auftraggeber und einem Auftragnehmer fest.

Eindeutigkeit von Anforderungen

Damit eine Anforderung in weiteren Entwicklungsschritten einfacher handhabbar ist und zu weniger Problemen führt, muss diese gewisse Qualitätskriterien erfüllen. Gute Anforderungen sind eindeutig und können nur in genau einer Art und Weise verstanden werden. Die folgende Anforderung ist nicht eindeutig:

> Bei Auftreten eines Fehlers im System muss die Warn-LED farbig leuchten.

In diesem Beispiel ist nicht definiert, bei welcher Art von Fehler die Warn-LED leuchten soll. Zudem ist nicht ersichtlich, welche Farbe die LED haben soll (z.B. rot bei gravierenden Fehlern, gelb bei weniger gravierenden Fehlern). Wird dies nicht weiter festgelegt, führt diese Anforderung entweder zu mehr Kommunikationsaufwand in der Entwicklung oder einem Endprodukt, welches nicht den Erwartungen entspricht. In beiden Fällen entstehen Mehrkosten.

Konsistenz von Anforderungen

Anforderungen müssen untereinander konsistent sein. Das heißt, es dürfen keine zwei Anforderungen existieren, die sich gegenseitig widersprechen.

> Die Warn-LED muss gelb leuchten, wenn das System einen Fehler der Fehlerklasse 1 erkannt hat.

> Die Warn-LED darf nur genau dann leuchten, wenn das System einen Fehler der Fehlerklasse 2 oder höher erkannt hat.

Diese Anforderungen widersprechen sich, da kein System entwickelt werden kann, welches beide Anforderungen erfüllt.

Praktikabilität von Anforderungen

Weiterhin müssen Anforderungen praktikabel[2] sein. Die Umsetzung muss mit dem aktuellen Stand der Technik möglich sein, es dürfen keine größeren Technologiefortschritte notwendig sein.

> Zur Fehlerprävention muss die Warn-LED bereits leuchten, wenn ein Fehler erst innerhalb der nächsten Minute auftreten wird.

Da der genaue Zeitpunkt des Eintretens eines Fehlers in der Regel nicht vorhersehbar ist, kann diese Anforderung nicht umgesetzt werden. Eine Menge von Anforderungen, die diese Kriterien erfüllt, kann beispielsweise wie folgt formuliert werden:

2 Engl. feasible

Die Warn-LED muss gelb leuchten, wenn ein Fehler der Fehlerklasse 1 erkannt wurde.

Die Warn-LED muss rot leuchten, wenn ein Fehler der Fehlerklasse 2 oder höher erkannt wurde.

Die Warn-LED muss erlöschen, wenn der Fehler nicht mehr auftritt.

Im ISO-Standard 29148 [52] werden weitere Merkmale von guten Anforderungen definiert: notwendig, lösungsneutral, vollständig, singulär, nachverfolgbar und überprüfbar. Diese werden in Abschnitt 2.1.2 im Detail beschrieben.

1.2 ANFORDERUNGEN UND ZUSATZINFORMATIONEN

Anforderungsdokumente enthalten neben Anforderungen weitere Informationen, sogenannte Zusatzinformationen. Dazu zählen unter anderem:

GROBE FUNKTIONSBESCHREIBUNG: Wird in einem Lastenheft eine Funktion spezifiziert, so wird zu dieser Funktion meist auch eine grobe Funktionsbeschreibung verfasst. In dieser Beschreibung wird eine Funktion auf sehr hohem Abstraktionsniveau beschrieben. Diese Beschreibungen sind hilfreich, um eine Funktionsspezifikation besser verstehen und einordnen zu können.

ANMERKUNGEN: Wird zum Verständnis einer Anforderung zusätzliches Hintergrundwissen benötigt, kann dieses in einer Anmerkung zur Anforderung abgelegt werden.

BEGRÜNDUNGEN: Wird eine Anforderung auf Basis einer Entscheidung auf eine gewisse Art und Weise formuliert, kann eine zugehörige Begründung als Zusatzinformation abgelegt werden.

BEISPIELE: Ist eine Anforderung schwer verständlich, können zusätzlich Beispielszenarien beschrieben werden, die verdeutlichen, für welche Situationen eine bestimmte Anforderung gedacht ist.

ABBILDUNGEN: Abbildungen können genutzt werden, um nur schwierig beschreibbare Sachverhältnisse besser zu dokumentieren.

REFERENZEN: Wird beispielsweise eine Schnittstelle oder eine von mehreren Systemen verwendete Funktion beschrieben, kann auf andere Anforderungsdokumente verwiesen werden, in denen diese Funktion detailliert spezifiziert ist.

Diese Informationen dienen in der Entwicklung mehreren Zwecken. Zum einen werden dadurch die Anforderungen dokumentiert. Somit kann später besser nachvollzogen werden, warum bestimmte Anforderungen existieren. Zum anderen kann ein Zulieferer die Dokumente besser verstehen und der Kommunikationsaufwand und das Risiko von Missverständnissen wird reduziert.

Alle diese Inhalte sind keine Anforderungen. Dies hat Auswirkungen auf Prozesse und Dokumente, die auf Anforderungsdokumenten aufbauen. So müssen beispielsweise in Testspezifikationen nur Anforderungen durch Testfälle abgedeckt werden. Wie auch in der Softwareentwicklung kann im Anforderungsmanagement die Testfallabdeckung von Anforderungsdokumenten gemessen werden. Diese berücksichtigt nur Anforderungen, jedoch nicht Zusatzinformationen.

Implementiert ein Zulieferer ein System, so muss der Zulieferer nur Anforderungen berücksichtigen, nicht jedoch die zusätzlichen Informationen. Üblicherweise werden die Anforderungen auch als Grundlage zur Abrechnung genutzt. Änderungen an Anforderungen werden ab einem gewissen Zeitpunkt sehr teuer, Informationen können jedoch beliebig hinzugefügt und entfernt werden.

Das Beispiel von oben kann wie folgt um Zusatzinformationen (kursiv) erweitert werden:

> *Die Funktion Warnung signalisiert dem Anwender über eine LED, wenn die Funktionalität des Gesamtsystems durch einen Fehler eingeschränkt ist.*
>
> Die Warn-LED muss gelb leuchten, wenn ein Fehler der Fehlerklasse 1 erkannt wurde.
>
> Die Warn-LED muss rot leuchten, wenn ein Fehler der Fehlerklasse 2 oder höher erkannt wurde.
>
> *Mögliche Fehler sind im Dokument „Systemübergreifende Fehlerspezifikation“, Abschnitt „Fehlerklassen“ spezifiziert.*
>
> Die Warn-LED muss erlöschen, wenn die Ursache des Fehlers behoben wurde.
>
> *Erreicht beispielsweise die Motortemperatur einen kritischen Schwellenwert (Fehlercode xyz, Fehlerklasse 3), leuchtet die Warn-LED rot. Erst wenn die Temperatur wieder unter den Schwellenwert sinkt, erlischt die Warn-LED.*

Durch diese Erweiterungen hat sich die Spezifikation nicht geändert, da lediglich Informationen ergänzt wurden. Allerdings sind die Anforderungen deutlich besser nachvollziehbarer. Der Kontext der Funktion ist durch die grobe Funktionsbeschreibung und das abschließende Beispiel klarer. Ebenfalls ist klar, an welcher Stelle wichtige Begriffe (Fehler, Fehlerklasse) spezifiziert werden.

Es ist vorteilhaft, während der Entwicklung von Systemen beim Verfassen der Anforderungsdokumente nicht nur darauf zu achten, dass

Anforderungen den Qualitätskriterien genügen, sondern auch durch angemessen viele Zusatzinformationen soweit beschrieben werden, dass deren Bedeutung klar ist.

Dazu ist in Lastenheften eine saubere Trennung zwischen Anforderungen und Zusatzinformationen notwendig. Um diese zu erreichen, muss zum einen eine technische Möglichkeit zur Markierung von Anforderungen und Informationen gegeben sein. Viel kritischer ist jedoch, dass Entwickler während der Dokumentation von Anforderungen auf eine saubere Trennung zwischen Anforderung und Information achten müssen.

1.3 ANFORDERUNGSQUALITÄT IN DER REALITÄT

Oftmals ist dies in Lastenheften jedoch nicht der Fall. Beobachtungen in Anforderungsdokumenten haben ergeben, dass Informationen oftmals nicht als solche deklariert und daher als Anforderung gewertet werden und Informationen nicht sauber von Anforderungen getrennt werden.

In der Realität sieht das Beispiel von oben wie folgt aus:

> Die Warn-LED leuchtet gelb, wenn ein Fehler der Fehlerklasse 1 erkannt wurde. Mögliche Fehler sind im Dokument „Systemübergreifende Fehlerspezifikation“, Abschnitt „Fehlerklassen“ spezifiziert.
>
> Die Warn-LED leuchtet rot, wenn ein Fehler der Fehlerklasse 2 oder höher erkannt wurde.
>
> Die Warn-LED muss erlöschen, wenn der Fehler nicht mehr auftritt, beispielsweise wenn die Motortemperatur wieder unterhalb eines Maximalwerts liegt.

Diese Art der Dokumentation von Anforderungen erschwert es erheblich, mit den Anforderungen zu arbeiten. Das Problem kann auf unterschiedliche Arten angegangen werden.

Zum einen kann von vornherein auf eine saubere Trennung von Anforderungen Informationen geachtet werden, indem Entwickler gezielt geschult werden, Anforderungen entsprechend zu dokumentieren. Durch den Einsatz von Boiler-Plates [8] wie beispielsweise EARS[3] [78] können Richtlinien zur Notation von Anforderungen teilweise erzwungen werden. So wird implizit dafür gesorgt, dass Anforderungen keine unnötigen Zusatzinformationen enthalten. Diese müssen dann explizit als separate Objekte dokumentiert werden.

In großen Konzernen sind diese Techniken allerdings nur mit Einschränkungen anwendbar. Es existieren bereits viele Lastenhefte. Werden neue Systeme und Funktionen entwickelt, so werden die dazuge-

3 Easy Approach to Requirements Syntax

hörigen Lastenhefte meistens nicht neu erstellt, sondern aus bestehenden Lastenheften von Vorgängersystemen und -funktionen abgeleitet. Wurde in diesen älteren Lastenheften nicht auf eine saubere Trennung von Informationen und Anforderungen geachtet, werden diese Qualitätsmängel ebenfalls mit in die neuen Lastenhefte übernommen.

Die nachträgliche Korrektur ist sehr zeitaufwändig und nur im Rahmen von umfassenden Reviews möglich. Das Neuschreiben von mit Qualitätsmängeln behafteten Lastenheften ist in der Regel nicht möglich, da dafür nicht genügend Ressourcen zur Verfügung stehen.

Fehlerkorrektur durch Werkzeuge

Abhilfe schaffen Werkzeuge, mit denen Anforderungsdokumente automatisch auf Fehler überprüft werden können. Der Smell-Detektor von Femmer u. a. [37] sucht nach Schwachstellen in Anforderungsdokumenten und warnt den Autor, wenn vordefinierte Regeln durch einzelne Anforderungen verletzt werden. Dazu gehört zum Beispiel die Verwendung von *Weak-Words*. Automatische Änderungen werden nicht durchgeführt, da dies zum einen nur in seltenen Fällen ohne Veränderung des Anforderungsinhalts möglich ist. Zum anderen möchten Entwickler die volle Kontrolle über die Anforderungen behalten und wenden daher keine Werkzeuge an, die automatisch Änderungen an den Anforderungen durchführen.

1.4 TRENNEN VON ANFORDERUNGEN UND ZUSATZINFORMATIONEN

Auch die Trennung und Klassifikation von Anforderungen und Zusatzinformationen ist eine Aufgabe, die nur in seltenen Fällen automatisch durchgeführt werden kann. Dies ist in der Regel nur dann möglich, wenn durch einen Entwickler vergessen wurde, ein ordnungsgemäß formuliertes Objekt in einem Anforderungsdokument korrekt zu klassifizieren. Das obige Beispiel hat jedoch gezeigt, dass Textbausteine gegebenenfalls umformuliert werden müssen, um eine saubere Trennung herzustellen.

Ein Ansatz, welches Entwickler bei dieser Aufgabe unterstützt, muss demzufolge folgende Eigenschaften aufweisen:

ANALYSE BESTEHENDER ANFORDERUNGEN: Der Ansatz muss grundsätzlich in der Lage sein, bestehende Anforderungen zu analysieren, da insbesondere in großen Konzernen bereits viele Anforderungsdokumente existieren.

ART DER ANALYSE. Um eine möglichst hohe Qualität von Anforderungsdokumenten zu erreichen, muss der Ansatz in der Lage sein, sowohl falsch klassifizierte und unsauber formulierte Objekte als auch Objekte, in denen Anforderungen und Informationen vermischt sind, zu erkennen.

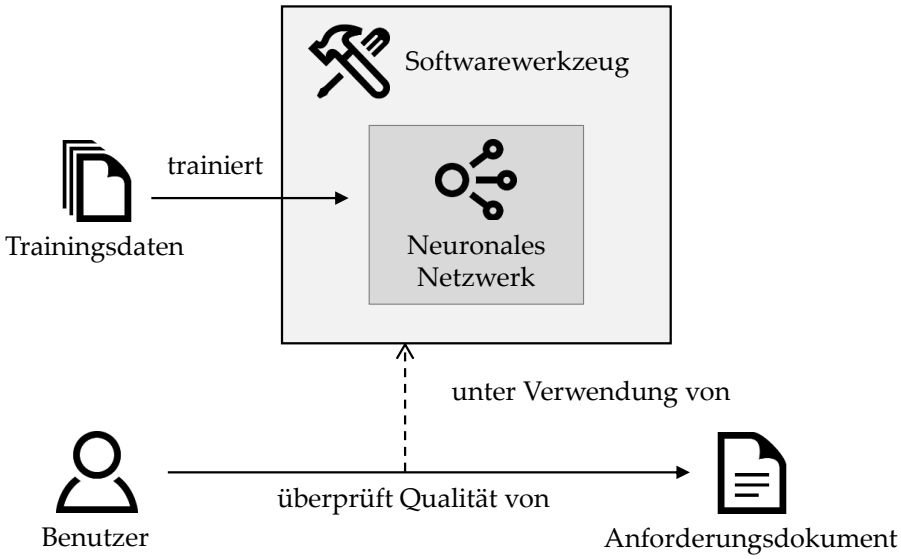


Abbildung 1.1: Überblick über den Ansatz

EINBEZIEHUNG DES BENUTZERS: Anforderungen dürfen durch den Ansatz nicht automatisch verändert werden. Vielmehr muss der Benutzer auf mögliche Qualitätsdefizite hingewiesen werden.

ART DER HINWEISE: Die ausgegebenen Hinweise müssen verständlich und schnell umsetzbar sein, damit Anwender zur Verbesserung ihrer Lastenhefte angeregt werden.

NACHHALTIGKEIT: Optimalerweise werden Anwender durch die Anwendung des Ansatzes lernen, beim Schreiben neuer Anforderungsdokumente gezielt auf die Trennung von Anforderungen und Informationen zu achten.

In dieser Arbeit wird ein Ansatz mit dem Ziel entwickelt, diese Punkte zu erfüllen. In Abbildung 1.1 ist dieser Ansatz grob dargestellt. Als zentraler Baustein wird ein künstliches neuronales Netz verwendet. Dieses ist durch Training auf bestehenden Anforderungsdokumenten darauf spezialisiert, in natürlichsprachlichen Sätzen Anforderungen von Zusatzinformationen zu unterscheiden. Da das verwendete Netzwerk Wahrscheinlichkeiten ausgibt, kann es ebenfalls verwendet werden, um unsauber formulierte Anforderungen zu identifizieren, da für diese weniger hohe Wahrscheinlichkeiten ausgegeben werden.

Der Ansatz wird in ein Werkzeug integriert, welches Benutzer auf Schwachstellen hinsichtlich der Klassifikation von Anforderungen und Informationen in Lastenheften hinweist. Ein Benutzer dieses Werkzeugs kann effizienter und effektiver Fehler in Lastenheften finden. Damit Anwender die vom Werkzeug gegebenen Hinweise besser und schneller verstehen können, wird zusätzlich ein spezielles Verfahren entwickelt, welches die Ursachen einer Entscheidung des Werkzeugs ermittelt und visualisiert.

Bestenfalls werden Entwickler durch die konsequente Anwendung des Werkzeugs ein besseres Bewusstsein für das Schreiben von Anforderungsdokumenten mit sauberer Trennung von Anforderungen und Zusatzinformationen entwickeln. So kann später viel Zeit gespart werden, da zeitaufwändige Reviews in ihrem Umfang erheblich reduziert werden können.

1.5 AUFBAU DER ARBEIT

Im Folgenden wird kurz der weitere Aufbau der Arbeit beschrieben.

Kapitel 2 erläutert die zum Verständnis notwendigen Grundlagen. Dazu gehören Grundlagen zum maschinellen Lernen, zu Text-Clustering-Verfahren, Textklassifikationsverfahren und neuronalen Netzwerken, insbesondere *Convolutional-Neural-Networks*. Es werden auch grundlegende Begriffe und Arbeitsabläufe des Anforderungsmanagements erläutert. In Kapitel 3 wird ein kurzer Einblick in die Anforderungsdokumente gegeben, die in dieser Arbeit verwendet wurden.

Die Konstruktion und das Training eines Klassifikators, der zwischen Anforderungen und Informationen unterscheiden kann, wird in Kapitel 4 vorgestellt. Dafür werden die Informationen über Anforderungsdokumente aus dem vorangegangenen Kapitel genutzt. In Kapitel 5 wird der Klassifikator erweitert. Zum einen wird gezeigt, wie die Genauigkeit des Klassifikators durch Einbezug von Kontextinformationen gesteigert werden kann, zum anderen wird ein Ansatz zur visuellen Erklärung von Netzwerkentscheidungen vorgestellt.

Der Ansatz wird in Kapitel 6 evaluiert. Zuerst wird untersucht, welchen Einfluss die unterschiedlichen Hyperparameter auf die Genauigkeit des Klassifikators haben und diejenigen Hyperparameter ausgewählt, welche die besten Ergebnisse erzielen. Der Ansatz wird in ein Werkzeug integriert. Anhand von zwei empirischen Studien wird gezeigt, dass ein durch das Werkzeug unterstütztes Review von Anforderungsdokumenten schneller und effektiver durchgeführt werden kann.

Kapitel 7 überträgt den Ansatz auf ein anderes Klassifikationsproblem und zeigt damit, dass die gewonnenen Erkenntnisse nicht nur auf die Klassifikation von Anforderungen und Informationen beschränkt sind. In Kapitel 8 werden verwandte Arbeiten vorgestellt. Kapitel 9 fasst die gewonnenen Erkenntnisse zusammen und diskutiert weitere Anknüpfungspunkte.

2 GRUNDLAGEN

In diesem Kapitel werden die zum Verständnis der Arbeit benötigten Grundlagen erläutert. Die einzelnen Abschnitte sind eigenständig und können mit den entsprechenden Vorkenntnissen übersprungen werden.

2.1 ANFORDERUNGSMANAGEMENT

Anforderungsmanagement (engl. „Requirements Engineering“, RE) bezeichnet „eine Disziplin, die sich mit allen Aspekten im Zusammenhang mit der systematischen Entwicklung einer vollständigen, konsistenten und eindeutigen Spezifikation beschäftigt, in der beschrieben wird, was ein softwaregestütztes Produkt leisten soll [...], und die als Grundlage für Vereinbarungen zwischen allen Stakeholdern dienen kann.“ [88]. Zu den im Anforderungsmanagement abgedeckten Aktivitäten zählen die Gewinnung und Dokumentation und Überprüfung der Übereinstimmung von Anforderungen und deren Management und Validierung [90]. Die durch das Anforderungsmanagement erzeugten Anforderungsartefakte bilden die Basis für weitere Schritte in der Entwicklung von Systemen.

Anforderungsmanagement ist von großer Relevanz, da die dabei entstehenden Artefakte großen Einfluss auf weitere Entwicklungsschritte und damit den Erfolg des Projektes haben. Im Chaos-Report von 2014 [92] und 2015 [106] wird anhand von Feldstudien und Umfragen gezeigt, dass lediglich 29% der erfassten Projekte erfolgreich abgeschlossen wurden. 52% wurden mit Zeit- oder Budgetüberschreitungen abgeschlossen und 19% sind gescheitert (Stand 2015). Als Ursachen für verzögerte oder fehlgeschlagene Anforderungen wurden als primäre Gründe unvollständige und sich während des Projekts ändernde Anforderungen genannt.

Definition Anforderung

Eine Anforderungsspezifikation ist eine Sammlung von Anforderungen an ein System. Bei Anforderungen wird grundsätzlich zwischen drei Arten von Anforderungen unterschieden [90]. *Funktionale Anforderungen* sind Anforderungen, die Funktionen und Verhalten eines Systems spezifizieren. Dazu zählen beispielsweise Anforderungen, die Eingaben und dazu erwartete Ausgaben eines Systems spezifizieren. *Qualitätsanforderungen* spezifizieren hingegen qualitative Eigenschaften (u.a. Verfügbarkeit, Bedienbarkeit, Ausfallsicherheit) eines Systems. *Rahmenbedingungen* sind Anforderungen an die Prozesse, mit denen ein System entwickelt wird.

Besonders in der Automobilindustrie dienen Anforderungsdokumente als rechtliche Grundlage. Fahrzeugkomponenten werden in der Regel durch Zulieferer auf Basis von Anforderungsdokumenten eines Automobilkonzerns produziert. Weisen diese Komponenten im Betrieb Fehlverhalten auf und führen zu Sach- oder Personenschäden, kann anhand der Spezifikationen identifiziert werden, ob die jeweilige Komponente unterspezifiziert ist (Fehler des Herstellers) oder das tatsächliche Verhalten von der Spezifikation abweicht (Fehler des Zulieferers).

2.1.1 Aktivitäten im Anforderungsmanagement

Die im Anforderungsmanagement durchgeführten Aktivitäten lassen sich in drei Kernaktivitäten (*Dokumentation*, *Gewinnung* und *Übereinstimmung*) und zwei Querschnittsaktivitäten (*Validierung* und *Management*) einordnen [90].

Gewinnung von Anforderungen

Während der Gewinnungsaktivität werden Anforderungen an das zu entwickelnde System durch Kommunikation zwischen denen an der Entwicklung beteiligten Stakeholdern erfasst. Ziel dieser Aktivität ist es, das inhaltliche Verständnis der Anforderungen zu verbessern. Insbesondere werden in dieser Aktivität auch Anforderungsquellen identifiziert. Innovative Anforderungen können beispielsweise nur durch intensive Kommunikation zwischen bestimmten Stakeholdern entdeckt werden.

Dokumentation von Anforderungen

Die gewonnen Anforderungen werden in der Dokumentationsaktivität in Anforderungsdokumenten dokumentiert beziehungsweise spezifiziert. Das Ziel ist es, die für ein Projekt gestellten Vorgaben zur Dokumentation von Anforderungen zu erfüllen. Im Vordergrund steht dabei die Dokumentation von Anforderungen, allerdings sollten auch weitere Informationen wie Entscheidungen und dazugehörige Begründungen dokumentiert werden. Als Dokumentationsform bieten sich mehrere Formen an, darunter natürliche Sprache und Modelle. Die jeweils gewählte Dokumentationsform hängt von den beteiligten Stakeholdern ab, da diese mit der Dokumentation der Anforderungen arbeiten müssen.

Übereinstimmung von Anforderungen

In der letzten Kernaktivität wird sichergestellt, dass die gewonnenen und dokumentierten Anforderungen den Wünschen und Vorstellungen aller beteiligten Stakeholder entsprechen. In dieser Aktivität werden in den Anforderungen Konflikte zwischen Stakeholdern identifiziert und aufgelöst.

Validierung von Anforderungen

Die Validierungsaktivität ist eine Querschnittsaktivität, da in dieser Aktivität verschiedene Aspekte des Anforderungsmanagement validiert werden. Zum einen werden die Anforderungsartefakte validiert: Nur Anforderungen, die fehlerfrei sind und gewissen Qualitätsstandards genügen, eignen sich zur erfolgreichen Durchführung von weiteren Aktivitäten wie Architekturentwurf, Implementierung und Test.

Anforderungsartefakte werden dabei gegen die Ziele validiert, die in den drei Kernaktivitäten verfolgt werden: Vollständigkeit, korrekte Dokumentation und Konfliktfreiheit. Weiterhin werden die Kernaktivitäten validiert. Dazu gehört beispielsweise die Sicherstellung der korrekten Durchführung dieser Aktivitäten.

Management im Anforderungsmanagement

Das Management ist die zweite Querschnittsaktivität im Anforderungsmanagement. Diese Aktivität beschäftigt sich mit organisatorischen Aspekten wie der Ablage von Anforderungsdokumenten in Datenbanken und der Planung und strukturierten Durchführung der anderen Aktivitäten im Anforderungsmanagement.

2.1.2 Qualität von Anforderungen

In der Validierungsaktivität werden Anforderungen auf Fehlerfreiheit überprüft und sichergestellt, dass diese gewissen Qualitätsstandards genügen. Die Kriterien anhand derer die Anforderungen geprüft werden, sind im Standard ISO-29148 „Systems and software engineering — Life cycle processes — Requirements engineering“ [52] definiert:

NOTWENDIG. Eine Anforderung definiert eine wichtige Eigenschaft des Systems. Wird sie entfernt, existiert ein Defizit, welches nicht durch andere Eigenschaften des Systems erfüllt wird. Weiterhin ist die Anforderung aktuell gültig und darf nicht durch neuere Anforderungen obsolet geworden sein.

LÖSUNGSNEUTRAL: Eine Anforderung gibt ausschließlich an, welche Eigenschaften ein System besitzen soll. Es darf nicht festgelegt werden, wie die Anforderung umgesetzt werden soll. Allerdings sind in bestimmten Situationen Hinweise, wie eine Anforderung umgesetzt werden kann, durchaus hilfreich. Diese sollten allerdings nicht als Anforderungen im Anforderungsdokument selbst, sondern in gesondert gekennzeichneten Dokumentenabschnitten oder in ergänzenden Dokumenten abgelegt werden.

EINDEUTIG: Anforderungen müssen so formuliert werden, dass sie von allen Stakeholdern auf nur genau eine Art und Weise verstanden werden können.

KONSISTENT: Anforderungen müssen untereinander konsistent sein. Es darf keine zwei Anforderungen geben, die widersprüchliche Eigenschaften definieren.

VOLLSTÄNDIG: Eine Anforderung enthält alle Informationen, die zur Umsetzung der Anforderung notwendig sind.

ATOMAR: Eine einzelne Anforderung beschreibt genau eine Eigenschaft eines Systems. Zwei verschiedene Eigenschaften müssen durch jeweils eine eigene Anforderung abgedeckt werden.

ANGEMESSEN: Die Umsetzung der Anforderung muss möglich sein. Das bedeutet zum einen, dass die Umsetzung der Anforderung technisch möglich ist und keine größeren Technologiefortschritte erfordert. Zum anderen bedeutet dies allerdings auch, dass die Umsetzung innerhalb des Projektrahmens möglich ist (Kosten, Zeit, rechtliche Bedingungen).

NACHVERFOLGBAR: Eine Anforderung muss nachverfolgbar sein. Damit ist gemeint, dass zu einer Anforderung stets klar ist, warum diese Anforderung existiert (Nachverfolgbarkeit nach oben) und welche weiteren Artefakte wie beispielsweise Testfälle oder Detailspezifizierungen zu dieser Anforderung existieren (Nachverfolgbarkeit nach unten).

TESTBAR: Wird eine Anforderung umgesetzt, muss am dabei entstehenden System nachweisbar sein, dass die Anforderung umgesetzt wurde.

Ein Anforderungsdokument, dessen Anforderungen diese Eigenschaften besitzen, gilt als qualitativ hochwertig. Es ist daher erstrebenswert, während der Entwicklung von Anforderungsdokumenten auf die Einhaltung dieser Standards und gegebenenfalls vorhandener, unternehmensspezifischer Richtlinien zu achten.

2.2 MASCHINELLES LERNEN

Maschinelles Lernen ist ein breites und seit längerer Zeit etabliertes Forschungsfeld und Teildisziplin der Informatik. In diesem Feld wird versucht, die Intelligenz des Menschen auf Maschinen zu übertragen. Maschinen sollen so Aufgaben bewältigen können, die mittels konventioneller Programmierung nur schwierig, zeitaufwändig oder gar nicht umgesetzt werden können. Aufgrund der Vielfalt der Herangehensweisen und Anwendungen existiert keine einheitliche Definition für Maschinelles Lernen. Eine oftmals zitierte Definition stammt von Arthur Samuel:

> Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.
>
> *– Arthur Samuel, 1959*

Nach dieser Definition ist Maschinelles Lernen ein Forschungsfeld, in welchem Probleme nicht durch explizites Programmieren eines Lösungsalgorithmus gelöst werden. Stattdessen lernt ein Computer selbstständig, das Problem zu lösen. Eine aktuelle und ebenfalls viel zitierte Definition für die Lösung eines Problems durch selbstständiges Lernen stammt von Tom Mitchel:

> A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.
>
> – *Tom Mitchel, 1997* [82]

Ein klassisches Beispiel ist das Spiel Dame. Wenn ein Computerprogramm bei der Aufgabe *Dame spielen (T)* seine Fähigkeit, *ein Dame-Spiel zu gewinnen (P)* verbessert, indem es *durch Spielen gegen sich selbst (E)* Erfahrung darüber sammelt, welche Züge zum Sieg und welche Züge zur Niederlage führen, *lernt* das Programm.

Zu den ersten Modellen des maschinellen Lernens gehört das Perzeptron (1958, [97]), welches ein Vorgänger von neuronalen Netzwerken ist. 1986 wurde der Backpropagation-Algorithmus [98] entwickelt, mit dem es erstmals möglich war, größere neuronale Netzwerke effizient zu trainieren. Daraufhin wurden neben anderen Modellen viele weitere Arten neuronaler Netzwerke für unterschiedliche Zwecke entwickelt.

Allgemeines Vorgehen im maschinellen Lernen

Die im maschinellen Lernen verwendeten Modelle werden meist auf Datensätzen trainiert. Dieser Vorgang heißt *Training*. Im Training werden die *Parameter* (auch *Gewichte*) des Modells anhand der im Datensatz vorhandenen *Beispiele* angepasst, um die Fähigkeit des Modells, eine vorgegebene Aufgabe zu lösen, zu verbessern. Dazu erlernt das Modell, Muster in den Eigenschaften (auch *Features*) der Beispiele zu erkennen, die sich in vielen Beispielen wiederholen. Die Menge aller Features, die an einem Beispiel ausgeprägt sein können, wird auch *Feature-Raum* genannt.

Vor dem Training müssen die Modelle erstellt werden. Dabei werden bei vielen Modellen *Hyperparameter* festgelegt. Diese Parameter geben Eigenschaften des Modells an (z.B. Größe, Anzahl Ebenen) und werden nicht trainiert, sondern durch Experten entsprechend der zu lösenden Aufgabe festgelegt. Nach dem Training eines Modells wird dieses verwendet, um die Aufgabe auf während des Trainings nicht gesehenen Beispielen durchzuführen. Dieser Vorgang heißt *Inferenz*.

Aufgaben im maschinellen Lernen

Maschinelles Lernen wird für viele unterschiedliche Aufgaben eingesetzt. Für jede dieser Aufgaben bieten sich jeweils andere Modelle an.

SUPERVISED-LEARNING: Modelle werden anhand vorgegebener Beispiele und zugehöriger Ausgaben darauf trainiert, die Ausgaben der Beispiele zu reproduzieren und vergleichbare Ausgaben bei ungesehenen Beispielen zu erzeugen. Bei Klassifikationsmodellen erlernt das Modell, Beispiele zu diskreten Klassen zuzuordnen. Anwendungsfalls: Klassifikation handschriftlicher Zahlen [69] und Erkennung von Spam [30]. Regressionsmodelle erlernen, Beispiele zu kontinuierlichen Werten zuzuordnen. Anwendungsfall: Vorhersage der Temperatur in Wettermodellen.

SEMI-SUPERVISED-LEARNING: Ähnlich zu Supervised-Learning werden auch hier Modelle anhand vorgegebener Beispiele und zugehöriger Ausgaben trainiert. Zusätzlich werden weitere Beispiele für das Training verwendet, zu denen die Ausgabe nicht bekannt ist. Die Modelle verwenden die Informationen aus den Beispielen ohne Ausgabezuordnung, um die Genauigkeit des Modells zu erhöhen.

UNSUPERVISED-LEARNING: Modelle werden auf vorgegebenen Beispielen trainiert, die jedoch keine zugeordnete Ausgabe besitzen. Die Modelle werden mit dem Ziel trainiert, Muster in den Beispielen zu erkennen. Ein Anwendungsfall von Unsupervised-Learning ist Clustering: Hier erlernen Modelle, ähnliche Beispiele in einem Datensatz zu gruppieren (z.B. Finden ähnlicher Dokumente in einem Dokumentenmanagementsystem).

SEQUENCE-TO-SEQUENCE-LEARNING: Bei dieser Form des Lernens werden Modelle trainiert, Sequenzen von Eingabedaten in Sequenzen von Ausgabedaten zu transformieren. Diese Modelle werden besonders zur maschinellen Übersetzung natürlicher Sprache verwendet.

REINFORCEMENT-LEARNING: Diese Modelle werden nicht durch eine vorgegebene Menge an Beispielen trainiert, sondern anhand einer Nutzenfunktion, welche die Ausgaben des Modells bewertet. Das Modell wird darauf trainiert, diese Nutzenfunktion zu maximieren. Anwendungsfall: Optimierung des Verkehrsflusses durch intelligente Steuerung von Verkehrssignalen [7].

2.3 TEXT-CLUSTERING

Text-Clustering [4] ist die Anwendung von Clustering-Algorithmen auf Textdaten. Clustering-Algorithmen sind Algorithmen, die in Datenmengen Gruppen von ähnlichen Datenpunkten finden. Die Ähnlichkeit zweier Datenpunkte in den Daten wird durch eine Ähnlichkeitsfunktion bestimmt. Clustering auf Text ist für viele Anwendungen interessant. Dazu gehören beispielsweise die automatische Organisation von Dokumenten in Kategorien, automatische Zusammenfassung von Datenmengen oder durch den Einsatz von Co-Training [85] auch die Klassifikation von Text.

Viele Algorithmen zum Clustering von Text sind allgemeine Algorithmen, die auf beliebige Daten angewendet werden können. Textdokumente können in Vektoren umgewandelt werden, wobei die Elemente der Vektoren das Vorhandensein oder die Abwesenheit bestimmter Wörter eines Wörterbuchs kennzeichnen (*Bag-of-Words*). Diese Techniken funktionieren aufgrund der folgenden speziellen Eigenschaften von Text in der Regel nicht sehr gut:

- Die Dimensionalität von Text ist in der Regel sehr groß, während einzelne Datenpunkte nur wenige der Dimensionen nutzen.
- Viele Wörter sind typischerweise miteinander verwandt. Die Anzahl an Konzepten („Principal Components“) in einem Wörterbuch ist in der Regel viel kleiner als das Wörterbuch selbst.
- Dokumente können unterschiedlich lang sein und Wörter unterschiedlich häufig auftreten. Daher müssen die Repräsentationen der Dokumente normalisiert werden.

Um auf Textdaten dennoch effektiv Clustering durchführen zu können, müssen daher verschiedene Vorkehrungen getroffen werden.

Feature-Auswahl und -Transformation

Im ersten Schritt werden aus den Features der Datenmenge für das Clustering relevante Features ausgewählt. Ein Feature ist in der Regel das Vorhandensein eines bestimmten Worts des Wörterbuchs im Dokument. Mithilfe der Dokumenthäufigkeit einzelner Wörter können besonders häufig auftretende Wörter (auch Stoppwörter genannt) und sehr selten auftretende Wörter entfernt werden. Diese Wörter tragen in der Regel nicht zur Bedeutung eines Dokuments bei, bzw. sind so selten, dass anhand dieser Wörter zwei Dokumente nicht ausreichend gut miteinander verglichen werden können.

Zur Normalisierung der Worthäufigkeiten wird oft die TF-IDF-Repräsentation („Term Frequency - Inverse Document Frequency“) eingesetzt. In dieser Repräsentation wird die Häufigkeit eines Worts in einem Dokument durch dessen inverse Dokumenthäufigkeit normalisiert. Dadurch wird die Bedeutung häufig vorkommender Wörter reduziert und die Bedeutung weniger häufig vorkommender Wörter, die in der Regel aussagekräftiger bzgl. der Bedeutung eines Dokuments sind, erhöht.

Verfahren wie Latent-Semantic-Indexing (LSI) und Principal-Component-Analysis (PCA) werden zur Reduzierung der Dimensionalität von Textdaten eingesetzt. Ziel dieser beiden Verfahren ist die Konstruktion eines kleineren Feature-Raums, in dem jedes Feature eine lineare Kombination mehrerer Features aus dem ursprünglichen Feature-Raum ist. Der reduzierte Feature-Raum ist in der Regel kleiner und rauschfreier und eignet sich damit besser zum Clustering von Textdaten.

Distanzbasierte Clustering-Algorithmen

Distanzbasierte Clustering-Algorithmen verwenden zur Bestimmung der Ähnlichkeit zweier Dokumente ein Textdistanzmaß. Das am weitesten verbreitete Maß ist die Kosinus-Ähnlichkeit. Dieses Maß berechnet den Kosinus des Winkels zweier Vektoren. Umso ähnlicher die Vektoren zweier Dokumente sind, desto geringer ist deren Winkel zueinander im Feature-Raum und desto größer demnach Kosinus. Weitere Maße zur Bestimmung von Textähnlichkeit sind die Levenshtein-Distanz und die Jaro-Winkler-Distanz.

Hierarchische Clustering-Algorithmen gruppieren Dokumente basierend auf deren Ähnlichkeit schrittweise in Cluster. Diese Algo-

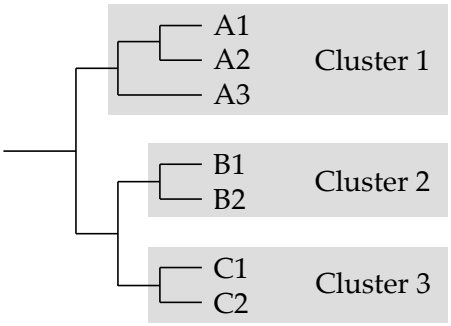


Abbildung 2.1: Dendrogramm einer hierarchischen Clusteranalyse

rithmen ordnen zu Beginn jedes Dokument einem eigenen Cluster zu und vereinigen solange ähnliche Cluster, bis alle verbleibenden Cluster gemäß des verwendeten Ähnlichkeitsmaßes genügend weit voneinander entfernt sind. Dabei entsteht eine Clusterhierarchie wie in Abbildung 2.1 gezeigt, auch Dendrogramm genannt. Darin sind die Blätter die Dokumente und die Knoten die gebildeten Cluster. Es existieren unterschiedliche hierarchische Cluster-Algorithmen, die sich in der Art und Weise, wie die Cluster gebildet werden, unterscheiden:

SINGLE-LINK-CLUSTERING: In Single-Link-Clustering wird die Ähnlichkeit zweier Gruppen von Dokumenten auf Basis der Ähnlichkeit der zwei ähnlichsten Dokumente in diesen beiden Gruppen bestimmt. Der große Vorteil dieses Verfahrens ist dessen Effizienz, da lediglich die Distanz aller Dokumentpaare bestimmt und sortiert werden muss. Ein Nachteil ist, dass Single-Link-Clustering durch wiederholtes Vereinigen von Dokumentgruppen zu Ketten ähnlicher Dokumente führen kann und so sehr breite Cluster entstehen.

GROUP-AVERAGE-LINK-CLUSTERING: In diesem Verfahren wird die Ähnlichkeit zweier Gruppen von Dokumenten auf Basis der durchschnittlichen Distanz aller Dokumentpaare dieser beiden Gruppen bestimmt. Dieses Verfahren ist deutlich langsamer als Single-Link-Clustering, ist dafür aber robuster und nicht für den beschriebenen Ketteneffekt anfällig.

COMPLETE-LINK-CLUSTERING: Die Ähnlichkeit zweier Cluster in diesem Verfahren ist die kleinste Ähnlichkeit zweier Dokumente beider Cluster. Analog zu Group-Average-Link-Clustering vermeidet dieses Verfahren den Ketteneffekt, ist aber ähnlich berechnungsintensiv.

Weiterhin existieren noch andere Cluster-Algorithmen wie beispielsweise Partitionierungsalgorithmen (u.a. k-Medoid-Clustering und k-Means-Clustering), die Datenmengen in eine vorher festgelegte Anzahl an Clustern unterteilen und dichtebasierte Clustering-Algorithmen (z.B. DBSCAN [33]).

2.4 TEXTKLASSIFIKATION

Das Problem der Klassifikation im Allgemeinen [3] ist wie folgt definiert: Sei $D = X_1, ..., X_n$ ein Datensatz, wobei jedes Element dieser Menge mit einer Klasse aus einer Menge von k verschiedenen, diskreten Klassen annotiert ist. Der Datensatz wird zur Konstruktion eines Klassifikationsmodells verwendet, welches eine Assoziation der Features des Datensatzes mit den Klassen herstellt. Das Klassifikationsmodell kann daraufhin verwendet werden, vorher ungesehenen Elementen basierend auf dessen Features eine Klasse zuzuordnen.

Textklassifikation bezeichnet die Anwendung von Klassifikationsmodellen auf Textdaten. Zur Vektorisierung von Dokumenten kann wiederum das Bag-of-Words-Modell verwendet werden: Dokumente werden in Vektoren umgewandelt, wobei die Vektoren die Länge eines vorher generierten Wörterbuchs besitzen und jedes Element der Dokumentvektoren das Vorhandensein oder Fehlen eines bestimmten Worts aus dem Wörterbuch kennzeichnet.

Feature-Auswahl und -Transformation

Analog zum Text Clustering muss auch bei der Textklassifikation eine Auswahl und Transformation von Features durchgeführt werden, da die verwendeten Wörterbücher in der Regel sehr groß sind und die Menge der tatsächlich in einem Dokument verwendeten Wörter sehr klein ist. Neben den in Abschnitt 2.3 genannten Techniken sind die folgenden Techniken anwendbar.

Der *Gini-Index* $G(w)$ eines Worts w gibt die Aussagekraft eines Worts zur Klassifikation an. Mögliche Werte reichen von $1/k$ bis 1, wobei 1 aussagt, dass das Wort nur in Beispielen genau einer Klasse zu finden ist und $1/k$ aussagt, dass es in den Beispielen aller Klassen gleichermaßen häufig auftritt. Je höher der Gini-Index eines Worts, desto relevanter ist es für die Klassifikation. Eine ähnliche Metrik ist *Information-Gain*. Diese Metriken können verwendet werden, um für die Klassifikation irrelevante Features zu eliminieren und damit die generierten Modelle zu vereinfachen.

Überblick Klassifikationsmodelle

Zur Klassifikation wurde eine Vielzahl von Modellen entwickelt, von denen eine Auswahl vorgestellt wird. Dies sind allgemeine Klassifikationsmodelle, die nicht nur auf die Klassifikation von Text beschränkt sind.

Entscheidungsbäume

Ein Entscheidungsbaum [93] ist eine anhand von Prädikaten erstellte hierarchische Ordnung auf den Elementen der Trainingsdatenmenge. Jeder Knoten dieser Ordnung teilt die Datenmenge anhand eines Prädikats in zwei Teile. Während des Trainings eines Entscheidungsbaums wird diese Unterteilung rekursiv solange durchgeführt, bis die Blätter des Baumes eine Mindestanzahl an Datenpunkten unterschreiten oder eine Bedingung an die Reinheit der Klassen innerhalb des Blattes erfüllen. Ein zu klassifizierendes Beispiel wird beginnend an der Wurzel mit den Prädikaten überprüft, bis ein Blatt erreicht

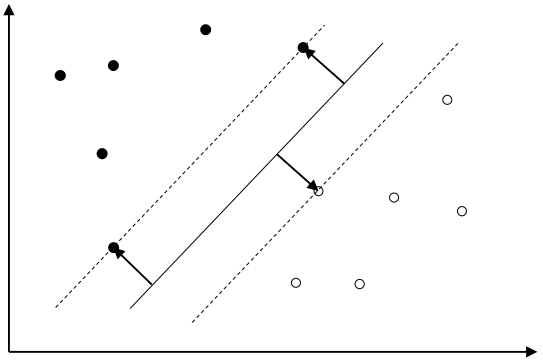


Abbildung 2.2: Lineare Separation bei Support-Vector-Machines

wird. Die in diesem Blatt vorrangig vorhandene Klasse bestimmt die Klassifikation des Beispiels.

In der Textklassifikation mit Entscheidungsbäumen sind diese Prädikate in der Regel das Vorhandensein oder Fehlen einzelner Wörter oder Wortgruppen. Allerdings sind noch viele weitere Arten von Prädikaten möglich. Sehr weit verbreitet ist der C5-Algorithmus [71], der den Entscheidungsbaum anhand einzelner Features aufbaut und dabei jeweils die Features mit dem größten *Information-Gain* auswählt.

Regelbasierte Klassifikatoren

Entscheidungsbäume sind eine spezielle Form von regelbasierten Klassifikatoren. Modelle von regelbasierten Klassifikatoren bestehen generell aus einer Menge von Regeln, die jeweils eine Bedingung im Feature-Raum auf eine Klasse abbilden. Während des Trainings werden diese Regeln aus dem Datensatz abgeleitet. Um unbekannte Beispiele zu klassifizieren, werden zuerst alle Regeln auf das Beispiel angewendet. Die Klasse des Beispiels ergibt sich aus der Mehrheit der Klassen, die von zutreffenden Regeln zurückgegeben wurden. Bei der Klassifikation von Text stellen diese Regeln meist Bedingungen an das Vorhandensein bestimmter Wörter im Dokument (z.B. $\mathtt{Honda} \,\&\, \mathtt{Toyota} \Rightarrow \mathtt{Cars}$).

Eine häufig verwendete regelbasierte Technik ist RIPPER [28]. RIPPER benutzt unter anderem *Information-Gain*, um häufig mit bestimmten Klassen verwendete Wörter und Wortgruppen zu identifizieren und konstruiert darauf basierend einen Regelsatz. Diese Technik ist besonders auf kleinen Datenmengen sehr effektiv.

Support-Vector-Machines

Support-Vector-Machines [54, 67] sind lineare Klassifikatoren. Das Modell linearer Klassifikatoren basiert auf einer Kombination aller Features mit linearen Koeffizienten. Eine mögliche Interpretation linearer Klassifikatoren ist eine Hyperebene im Feature-Raum, anhand derer eine Unterteilung der Beispiele in zwei Kategorien vorgenommen wird. Support-Vector-Machines versuchen eine Hyperebene zu finden, mit der eine möglichst gute Separation zweier Klassen erreicht wird. Abbildung 2.2 veranschaulicht dies. Dazu wird während des Trainings einer Support-Vector-Machine nach denjenigen Support-Vektoren gesucht, die eine Hyperebene mit der bestmöglichen Separation aufspannen.

Mithilfe des Kernel-Tricks [99] können Support-Vector-Machines auch nichtlineare Hyperebenen zur Separation zweier Klassen finden. Dazu wird der Feature-Raum durch eine nichtlineare Operation in einen neuen Feature-Raum transformiert, in welchem eine lineare Hyperebene zur Separation gesucht wird. In der Praxis werden aber insbesondere bei der Klassifikation von Text lineare Support-Vector-Machines aufgrund ihrer geringen Komplexität und einfachen Interpretierbarkeit eingesetzt und erreichen dennoch gute Ergebnisse.

Um mittels Support-Vector-Machines auch Klassifikationsprobleme mit mehr als zwei Klassen zu erlernen, wird das Klassifikationsproblem auf mehrere binäre Klassifikationsprobleme reduziert. Für jede Klasse wird eine Support-Vector-Machine trainiert, die jeweils eine Klasse von allen anderen Klassen separiert. Dieses Vorgehen heißt „One-versus-the-rest“ [21, p. 182].

Regressionsbasierte Klassifikatoren

Regressionsbasierte Klassifikatoren benutzten zur Klassifikation Regression. Regression wird normalerweise zur Modellierung von Relationen zwischen Eingabewerten und kontinuierlichen Ausgabewerten benutzt, kann aber durch die Festlegung einer Entscheidungsgrenze auch zur binären Klassifikation eingesetzt werden. Logistische Regression ist ein solcher Klassifikator und gehört ebenfalls den linearen Klassifikatoren an. Eine beliebige Menge von Features x wird linear mit Gewichten w kombiniert und daraufhin durch die Sigmoid-Funktion in eine Wahrscheinlichkeit y umgerechnet:

$$\begin{aligned} f(x) &= x_1 w_1 + x_2 w_2 + ... + x_n w_n \\ y &= sig(f(x)) = \frac{1}{1 + e^{-f(x)}} \end{aligned} \tag{2.1}$$

Die Gewichte w werden durch numerische Minimierung einer Kostenfunktion ermittelt, sodass y für die Beispiele einer Klasse im Datensatz groß und für die Beispiele der anderen Klasse klein ist. Ein ungesehenes Beispiel wird klassifiziert, indem y für dieses Beispiel berechnet wird. ist y größer als ein definierter Schwellenwert (in der Regel 0,5), so gehört es der einen Klasse an, ansonsten der Anderen.

Generative Klassifikationsmodelle

Die bisher vorgestellten Klassifikatoren sind diskriminative Modelle, d.h. sie berechnen eine Klasse oder Klassenwahrscheinlichkeit y basierend auf einer Beobachtung X im Feature-Raum: $P(y|X)$. Generative Modelle berechnen die Wahrscheinlichkeit einer Beobachtung gegeben einer Klasse: $P(X|y)$. Diese Wahrscheinlichkeit wird meist durch eine Kombination von Wahrscheinlichkeiten einzelner Features gegeben der Klasse y berechnet. Ein ungesehenes Dokument wird klassifiziert, in dem die Wahrscheinlichkeit des Auftretens dieses Dokuments für jede Klasse berechnet wird und diejenige Klasse mit der höchsten Wahrscheinlichkeit ausgewählt wird.

Das am häufigsten anzutreffende generative Klassifikationsmodell ist *Naïve-Bayes*. Dieses Modell nimmt an, dass sämtliche Features des

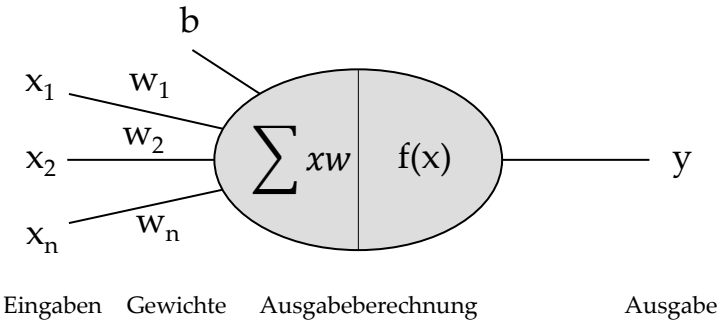


Abbildung 2.3: Modell eines künstlichen Neurons

Feature-Raums voneinander unabhängig sind und berechnet die Wahrscheinlichkeit eines Dokuments als Produkt der Wahrscheinlichkeiten aller Wörter innerhalb dieses Dokuments. Die Wahrscheinlichkeit $P(x|y)$ eines bestimmten Worts wird aus den Trainingsdaten gewonnen.

Die Annahme über die Unabhängigkeit einzelner Wörter trifft bei natürlicher Sprache in der Regel nicht zu. Dennoch erreichen Naïve-Bayes-Klassifikatoren sehr gute und mit anderen Klassifikatoren vergleichbare Ergebnisse.

Andere Klassifikationsmodelle

Neben den vorgestellten Klassifikatoren existieren viele weitere Ansätze zur Klassifikation, die aus Platz- und Zeitgründen nicht alle in dieser Arbeit vorgestellt und verglichen werden können. Neuronale Netzwerke sind ein weiterer Typ von Klassifikationsmodellen und werden im folgenden Abschnitt vorgestellt.

2.5 NEURONALE NETZWERKE

Neuronale Netzwerke [48] sind eines von vielen mathematischen Modellen, die im maschinellen Lernen eingesetzt werden. Mit neuronalen Netzwerken können viele Aufgaben wie Klassifikation, Clustering und Sequence-2-Sequence-Learning bewältigt werden. In den frühen Anfängen des maschinellen Lernens wurde versucht, die Vorgänge im Gehirn zu imitieren und künstliche neuronale Netzwerke zu erschaffen. McCulloch und Pitts [79] beschreiben dazu ein Modell, welches den biologischen Neuronen ähnlich ist.

Künstliche Neuronen

Abbildung 2.3 zeigt dieses Modell. Ein künstliches Neuron besitzt mehrere Eingänge und einen Ausgang. Abhängig von den Eingaben kann ein künstliches Neuron aktiviert werden und einen großen Wert ausgeben, oder inaktiv bleiben und einen kleinen Wert ausgeben. Ein künstliches Neuron trifft auf Basis von Eingaben eine Entscheidung und gibt diese aus. Ein einfaches Neuron ist wie auch eine Support-Vector-Machine ein lineares Klassifikationsmodell.

Die Entscheidung wird durch Gewichte gesteuert. Jeder Eingang eines Neurons besitzt ein festgelegtes Gewicht. Die Ausgabe eines

künstlichen Neurons wird berechnet, indem jede Eingabe mit dem Gewicht des entsprechenden Eingangs multipliziert und diese Zwischenergebnisse und ein Bias summiert werden:

$$y = f(\sum_i w_i x_i + b) \tag{2.2}$$

Aktivierungsfunktionen

f ist eine beliebige Aktivierungsfunktion. Im Modell gemäß McCulloch und Pitts ist dies die Threshold-Funktion, d.h. die Ausgabe des Neurons ist 1, wenn die gewichtete Summe größer als ein vorher festgelegter Wert ist, ansonsten 0. Andere Aktivierungsfunktionen sind möglich:

SIGMOID: Rechnet die Ausgabe eines Neurons in einen Wert zwischen 0 und 1 um. Diese Funktion wird häufig verwendet, um die Ausgabe eines Neurons in eine Wahrscheinlichkeit umzurechnen.

TANGENS HYPERBOLICUS: Diese Funktion hat ähnliche Eigenschaften wie die Sigmoid-Funktion und eignet sich daher ebenfalls als Aktivierungsfunktion.

RELU: Rectified-Linear-Units werden besonders bei neuronalen Netzwerken mit vielen Ebenen eingesetzt, da diese dem Problem des verschwindenden Gradienten entgegenwirken [43]:

$$\mathrm{ReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

Feed-Forward-Netzwerke

Mehrere künstliche Neuronen können kombiniert werden, um komplexere Entscheidungsprozesse zu modellieren. Sämtliche logische Operationen (AND, OR, NOT, NAND) können mithilfe von Neuronen nachgebildet werden [79]. Somit sind Netzwerke aus künstlichen Neuronen in der Lage, jede beliebige logische Funktion abzubilden.

Eine der einfachsten Architekturen von neuronalen Netzwerken sind Feed-Forward-Netzwerke (siehe Abbildung 2.4). In dieser Netzwerkarchitektur werden Neuronen in Ebenen angeordnet. Jedes Neuron einer Ebene ist mit allen Neuronen der darauffolgenden Ebene verbunden. Daher werden diese Ebenen auch *vollständig verknüpfte Ebenen* genannt. Die Neuronen der ersten Ebene geben lediglich die Eingaben des Netzwerks aus und besitzen keine Gewichte, daher ist die erste Ebene des Netzwerks die Eingangsebene. Die Neuronen der letzten Ebene geben das Ergebnis des Netzwerks aus. Zwischen diesen beiden Ebenen können beliebig viele versteckte Ebenen angeordnet werden. Jedes Neuron einer Ebene erkennt ein bestimmtes Muster in der vorangegangenen Ebene. Umso mehr versteckte Ebenen ein Feed-Forward-Netzwerk umfasst, desto komplexere Muster kann dieses Netzwerk erkennen.

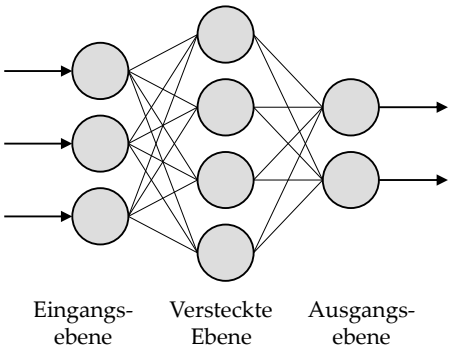


Abbildung 2.4: Feed-Forward-Netzwerk

Softmax-Aktivierungsfunktion

Eine weitere Aktivierungsfunktion, die häufig bei neuronalen Netzwerken bei den Neuronen der Ausgabeebene verwendet wird, ist die Softmax-Funktion:

$$\mathrm{softmax}(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}} \tag{2.4}$$

Analog zur Sigmoid-Funktion wird diese Funktion ebenfalls verwendet, um Ausgaben von Neuronen in Wahrscheinlichkeiten umzurechnen. Für die Ausgabe $y \in \mathbb{R}^n$ von n Ausgabeneuronen berechnet Softmax im Unterschied zu Sigmoid jedoch eine Wahrscheinlichkeitsverteilung über alle Neuronen. Das heißt, dass $\sum_{i=1}^{n} \mathrm{softmax}(y_i) = 1$.

Backpropagation

Rumelhart, Hinton und Williams [98] stellen einen Algorithmus vor („Backpropagation of Error"), der es erstmals ermöglicht, die Gewichte eines Netzwerks zur Lösung einer bestimmten Aufgabe automatisch anzupassen. Vorher war dies nur in Sonderfällen sehr eingeschränkt möglich oder wurde manuell erledigt.

Der Algorithmus vergleicht die Ausgabe eines Netzwerks bei einer bestimmten Eingabe mit der erwarteten Ausgabe und passt die Gewichte des Netzwerks solange schrittweise an, bis die tatsächliche Ausgabe mit der erwarteten Ausgabe übereinstimmt. Der Algorithmus arbeitet grob wie folgt:

1. Für das Netzwerk wird eine Kostenfunktion erstellt.

$$C(w, b) = \sum_{x \in X} (y_x - d_x)^2 \tag{2.5}$$

In dieser Funktion sind w und b die Gewichte des Netzwerks, X die Menge an Eingabe-Ausgabe-Paaren einer Trainingsmenge, y_x die tatsächliche Ausgabe für ein bestimmtes x und d_x die erwartete Ausgabe für ein bestimmtes x. Die tatsächliche Ausgabe des Netzwerks hängt ebenfalls von den Gewichten w und b ab. Das Ergebnis der Funktion ist umso größer, je stärker die Netzwerkausgabe von der erwarteten Ausgabe abweicht.

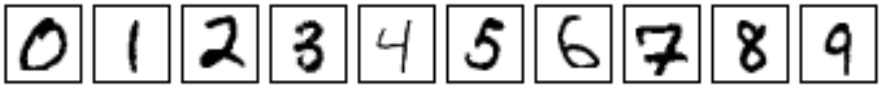


Abbildung 2.5: Beispiele aus der MNIST-Datenbank

2. Das Ziel ist es, das globale Minimum der Kostenfunktion C zu finden. Dazu wird für jede Variable (jedes Gewicht) die partielle Ableitung numerisch ermittelt. Das Ergebnis ist der Gradient δC der Funktion C.

3. Die Gewichte werden entsprechend ihrer partiellen Ableitung leicht modifiziert.

4. Schritt zwei und drei werden wiederholt, bis sich die Gewichte des Netzwerks nicht mehr ändern, d.h. der Gradient sehr klein ist und somit ein Minimum der Kostenfunktion gefunden wurde.

Dieses Verfahren zur Optimierung neuronaler Netzwerke wird auch „Stochastic Gradient Descent“ genannt. Auf der Basis von Backpropagation wurden weitere Verfahren zur Optimierung der Gewichte eines Netzwerks entwickelt, darunter RMSprop, Adam [58] und Adagrad [31].

MNIST

Das Standardbeispiel für neuronale Netzwerke ist das MNIST-Klassifikationsproblem. Bei diesem Problem muss ein Klassifikator in einem Bild handschriftliche Ziffern erkennen (Beispiele siehe Abbildung 2.5). Der Datensatz[1] enthält 60.000 beschriftete Trainingsbeispiele und 10.000 Testbeispiele, jeweils 28x28 Pixel groß.

Ein wie oben beschriebenes neuronales Netzwerk kann nun wie folgt erstellt und trainiert werden: Über 784 Eingangsneuronen wird ein zu klassifizierendes Bild in das Netzwerk Pixel für Pixel eingegeben. An zehn Ausgangsneuronen wird mittels Softmax eine Pseudo-Wahrscheinlichkeitsverteilung erzeugt.

Ohne versteckte Ebenen ist auf dem Testdatensatz eine Präzision von 85% möglich. Mit versteckten Ebenen kann die Genauigkeit der Klassifikation wesentlich erhöht werden. Mit einer versteckten Ebene (100 Neuronen) klassifiziert ein Feed-Forward-Netzwerk bereits 95% der Testbeispiele richtig. Moderne Netzwerkarchitekturen und Vorverarbeitungstechniken ermöglichen Genauigkeiten um 99,7% [26].

2.5.1 Unausgeglichene Datensätze

Datensätze, in denen eine Klasse häufiger vorkommt als andere Klassen, sind unausgeglichen [49]. Ein Datensatz, der Kreditkartentransaktionen enthält und zur Erkennung von Kreditkartenmissbrauch verwendet wird, wird nur sehr wenig Beispiele enthalten, in denen

1 http://yann.lecun.com/exdb/mnist/

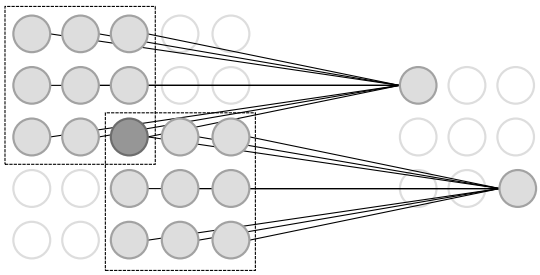


Abbildung 2.6: Zweidimensionale Convolution-Ebene

ein Missbrauch vorliegt. Ein Klassifikator, der auf diesem Datensatz trainiert wird, wird in der Regel erlernen, die vorrangig im Datensatz vorhandene Klasse zu erkennen und die unterrepräsentierte Klasse zu ignorieren. Um dem entgegenzuwirken, müssen Techniken zum Ausgleich des Datensatzes angewendet werden. Die folgenden vier Techniken werden üblicherweise bei neuronalen Netzwerken eingesetzt:

OVERSAMPLING: Bei diesem Verfahren werden die Beispiele der unterrepräsentierten Klasse dupliziert, bis der Datensatz ausgeglichen ist.

UNDERSAMPLING: Aus den Beispielen der überrepräsentierten Klasse werden zufällig Beispiele entfernt, bis der Datensatz ausgeglichen ist.

GENERATIVE ANSÄTZE: Auf Basis der vorhandenen Beispiele der unterrepräsentierten Klasse werden neue, künstliche Beispiele generiert, die den vorhandenen ähnlich sind. Eine sehr bekannte Technik ist SMOTE [25].

GEWICHTETES TRAINING: Während des Trainings wird der Beitrag jedes Beispiels zur Kostenfunktion um einen Faktor x erhöht, wobei x der inverse Anteil der Klasse des jeweiligen Beispiels im Datensatz ist. Umso seltener die Klasse, desto größer ist x.

2.6 CONVOLUTIONAL-NEURAL-NETWORKS

Neben Feed-Forward-Netzwerken sind Convolutional-Neural-Networks (CNN) eine weitere Form neuronaler Netzwerke [68, 69]. Diese wurden speziell für Klassifikation von Bildern entwickelt, da beispielsweise Feed-Forward-Netzwerke hier einen entscheidenden Nachteil haben. Im Gegensatz zu Feed-Forward-Netzwerken sind CNNs in der Lage, erlernte Muster an beliebigen Positionen in einem Bild zu erkennen, während erlernte Muster in Feed-Forward-Netzwerken stets an bestimmte Eingabeneuronen gebunden sind.

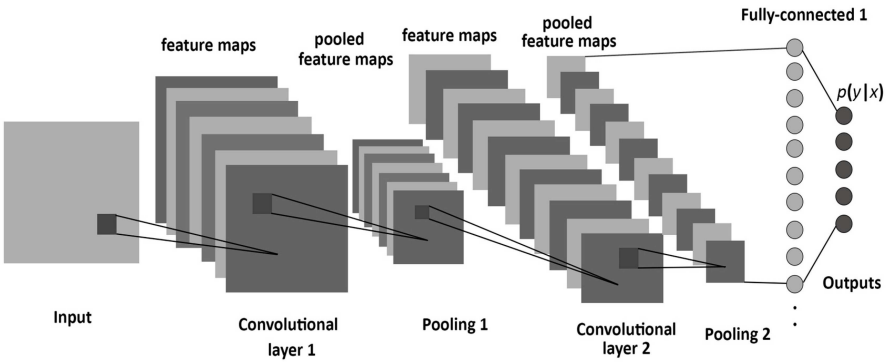


Abbildung 2.7: Deep Convolutional Neural Network [6]

Convolution in CNNs

In CNNs werden dazu Convolution-Ebenen eingesetzt. Im Gegensatz zu den vollständig verknüpften Ebenen von Feed-Forward-Netzwerken werden die Neuronen einer Ebene mit jeweils nur einer Teilmenge der Neuronen der vorherigen Ebene verbunden. In Abbildung 2.6 ist ein Neuron der Convolution-Ebene mit neun Neuronen der vorangegangenen Ebene verbunden. Weiterhin werden in einer Convolution-Ebene die Gewichte aller Neuronen geteilt. Die Berechnung der Ausgabewerte der beiden in Abbildung 2.6 hervorgehobenen Neuronen erfolgt zwar unter Verwendung verschiedener Neuronen der vorangegangenen Ebene, multipliziert werden deren Ergebnisse jedoch mit denselben Gewichten. Eine Convolution-Ebene ist somit in der Lage, ein bestimmtes Muster vordefinierter Größe (auch *Filter* genannt, im Beispiel 3x3) unabhängig von dessen Position in der vorangegangenen Ebene zu erkennen. Die Filter werden wie ein Fenster über die Eingabedaten geschoben und an jeder möglichen Position angewendet. Die Ausgabe einer Convolution-Ebene zeigt, an welchen Positionen in der vorangegangenen Ebene das Muster erkannt wurde.

In CNNs werden pro Convolution-Ebene mehrere Filter parallel eingesetzt, um gleichzeitig verschiedene Muster erkennen zu können (siehe Abbildung 2.7). Je Filter entsteht als Ergebnis eine *Feature-Map*. Wie auch bei Feed-Forward-Netzwerken können mehrere dieser Ebenen kombiniert werden, um komplexere Muster in den Eingabedaten zu erkennen.

Pooling in CNNs

In aktuellen Netzwerkarchitekturen werden zudem *Pooling-Ebenen* eingesetzt, mit denen die Größe der Feature-Maps reduziert wird. Meistens wird Max-Pooling verwendet: Aus einer Gruppe benachbarter Neuronen (*Kernel*) wird jeweils nur das größte Ergebnis ausgewählt. Bei einer Kernel-Größe von 2x2 wird die Höhe und Breite der Feature-Maps jeweils halbiert. Nach Convolution und Pooling werden wie auch bei Feed-Forward-Netzwerken vollständig verknüpfte Ebenen eingesetzt, um die Ausgaben der letzten Pooling-Ebene mit den Ausgabe-Neuronen zu verbinden.

Anwendungsbeispiele für CNNs

Anwendung finden Convolutional-Neural-Networks bei der Bilderkennung. Beispielsweise können CNNs die Testdaten des MNIST-

Klassifizierungsproblems mit einer Genauigkeit von 99,77% klassifizieren [26]. Insbesondere die Verwendung von Grafikprozessoren [104] erhöhte die Popularität von CNNs stark, da diese im Vergleich zu herkömmlichen CPU-Implementierungen die zum Training benötigte Zeit um den Faktor zwei bis 24 reduzieren.

2.7 WORD-EMBEDDINGS

Die bisher vorgestellten Textklassifikatoren benutzten zur Repräsentation von Dokumenten das Bag-of-Words-Modell. Dieses Modell besitzt einige Nachteile, die anhand der folgenden beiden Beispielsätze veranschaulicht werden:

- Das Wetter ist gut.
- Das Wetter ist schön.

Basierend auf dem Wörterbuch (Das, Wetter, ist, gut, schön) werden diese beiden Sätze jeweils in die Vektoren $[1\,1\,1\,1\,0]$ und $[1\,1\,1\,0\,1]$ umgewandelt. In dieser Darstellung wird jedes Wort unterschiedlich behandelt, obwohl die beiden Wörter gut und schön in diesen Sätzen etwa die gleiche Bedeutung haben.

Eigenschaften von Word-Embeddings

Word-Embeddings verfolgen einen anderen Ansatz. Word-Embeddings weisen jedem Wort w eines Wörterbuchs einen Vektor $\mathtt{emb}(w) \in \mathbb{R}^n$ zu. Die mit der Word-Embeddings-Implementierung Word2Vec generierten Vektoren weisen folgende Eigenschaften auf [80]:

- Die Vektoren von in ähnlichen Kontexten verwendeten Wörtern zeigen in ähnliche Richtungen. Die Cosinus-Ähnlichkeit der Vektoren der beiden Wörter gut und schön aus dem obigen Beispiel ist beispielsweise deutlich größer als die Ähnlichkeit zweier anderer Wörter dieser beiden Beispiele.
- Auf Word-Embeddings können arithmetische Operationen durchgeführt werden. Beispielsweise ist der Vektor $\mathtt{emb}(\mathtt{König}) + \mathtt{emb}(\mathtt{Frau}) - \mathtt{emb}(\mathtt{Mann})$ bei auf einem ausreichend großen Korpus trainierten Word-Embedding sehr ähnlich zum Vektor $\mathtt{emb}(\mathtt{Königin})$.

Die Wortvektoren können verwendet werden, um die Wörter eines Dokuments in Vektoren und das Dokument dementsprechend in eine Matrix konkatenierter Wortvektoren zu transformieren. Diese Dokumentmatrix kann daraufhin als Eingabe für Klassifikationsmodelle verwendet werden.

Trainieren von Word-Embeddings mit dem Skip-Gram-Modell

Word-Embeddings werden in der Word2Vec-Implementierung durch ein Skip-Gram-Modell auf einem möglichst großen Korpus natürlicher Sprache trainiert. Dieses Modell ist ein neuronales Netzwerk mit Feed-Forward-Architektur, welches die Wahrscheinlichkeit

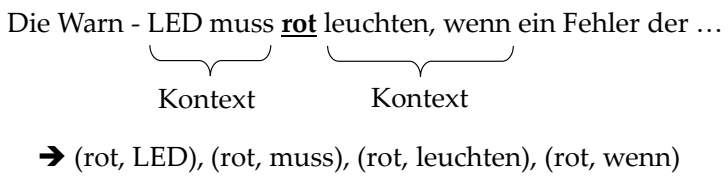


Abbildung 2.8: Bildung der Trainingspaare für das Skip-Gram-Modell

modelliert, dass ein bestimmtes Wort in der Nähe eines anderen Worts verwendet wird. Die Eingabe- und Ausgabeebene des Netzwerks umfassen so viel Neuronen wie das Wörterbuch des Korpus Einträge besitzt. Eine versteckte Ebene befindet sich dazwischen, dessen Anzahl an Neuronen auf Basis der Größe des Korpus festgelegt wird. Typisch sind Größen von 50 bis zu 500. Die Gewichte, die zur Berechnung der Ausgabeneuronen aus der versteckten Ebene verwendet werden, sind die gesuchten Word-Embeddings.

Das Modell wird mit Paaren von in der Nähe voneinander verwendeten Wörtern trainiert. Die Paare werden gebildet, indem für jedes Wort im Korpus Tupel mit Wörtern aus dem Kontext des Worts gebildet werden (siehe Abbildung 2.8). Die Größe des Kontexts, auch Fensterbreite genannt, wird vor dem Training festgelegt und bestimmt, wie spezifisch die trainierten Word-Embeddings sind.

GloVe: Global Vectors for Word Representation

Weitere Word-Embedding-Implementierungen existieren. GloVe [89] erzeugt Wortvektoren, die in Klassifikationsmodellen und anderen NLP-Aufgaben ähnlich einsetzbar sind wie die mit Word2Vec generierten Vektoren. GloVe trainiert die Vektoren nicht durch ein Modell, sondern zählt je Wort die Anzahl an unterschiedlichen Kontexten, in denen das Wort verwendet wird. Auf der entstehenden Matrix wird zur Gewinnung der Wortvektoren Dimensionsreduzierung durchgeführt.

2.8 TEXTKLASSIFIZIERUNG MIT CNNS

Die in Abschnitt 2.6 vorgestellten Convolutional-Neural-Networks lassen sich durch den Einsatz von Word-Embeddings auch auf die Klassifikation von Text übertragen [55, 57, 119]. Gegenüber anderen Klassifikationsmodellen, die das Bag-of-Words-Modell verwenden, bietet dieser Ansatz einige Vorteile. Durch den Einsatz von Word-Embeddings und Dokumentmatrizen zur Repräsentation von Eingabedokumenten bleibt die Reihenfolge der Wörter im Satz bestehen und kann in die Klassifikation mit einbezogen werden. Weiterhin besitzen in ähnlichen Kontexten verwendete Wörter ähnliche Wortvektoren. Somit ist es dem Modell möglich, Muster zu erlernen, in denen sich die Wörter von Fall zu Fall auch leicht unterscheiden können.

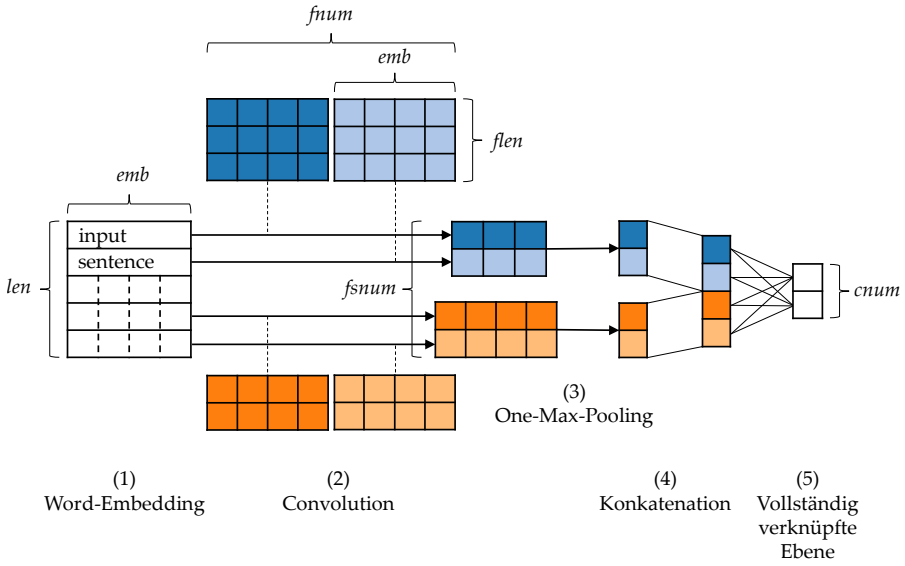


Abbildung 2.9: CNN-Netzwerkarchitektur zur Klassifikation von Text gemäß Kim [57]

Aufbau und Funktionsweise des Modells

Abbildung 2.9 zeigt den Aufbau eines CNNs, welches zur Klassifikation von Text entwickelt wurde. Der Aufbau und die Funktionsweise des Modells wird im Folgenden erläutert.

Zuerst wird ein zu klassifizierendes Dokument in eine Dokumentmatrix umgewandelt (1). Dazu werden die Wortvektoren aller Wörter des Eingabesatzes gestapelt. Die Wortvektoren werden vorher auf einem Korpus, der mindestens das Vokabular der zu klassifizierenden Dokumente enthält, trainiert. Für ein Eingabedokument der Länge $\mathtt{len}$ und ein Word-Embedding mit Vektoren der Länge $\mathtt{emb}$ entsteht so eine Dokumentmatrix $m \in \mathbb{R}^{\mathtt{emb},\mathtt{len}}$. Sind bestimmte Wörter des Eingabesatzes nicht im Vokabular des Word-Embeddings enthalten, werden diese nicht ignoriert, sondern entweder durch einen zufälligen Wortvektor oder durch einen vorher festgelegten Vektor für unbekannte Wörter ersetzt. Die Embedding-Operation ist bereits Teil des Netzwerks und die Vektoren des Word-Embeddings daher Gewichte, die während des Trainings angepasst werden können. In der Regel werden die Word-Embeddings während des Trainings allerdings nicht modifiziert.

Auf die Dokumentmatrix wird eine Convolution-Operation angewendet (2). Dabei wird eine feste Anzahl an Filtern $\mathtt{fnum}$ auf die Dokumentmatrix angewendet, um Feature-Maps zu berechnen. Jeder Filter wird während des Trainings ein bestimmtes Wortmuster erlernen. Die Breite der Filter ist die Größe des Word-Embeddings und die Länge $\mathtt{flen}$ bestimmt, wie viele Wörter die von den Filtern erlernten Muster umfassen können. Filter unterschiedlicher Längen können kombiniert werden, um es dem Netzwerk zu ermöglichen, Muster unterschiedlicher Längen zu erlernen. Je Filterlänge entsteht

so eine Feature-Map. Die Feature-Maps geben an, ob und an welchen Positionen im Dokument bestimmte Wortmuster erkannt wurden.

Auf die Feature-Maps wird eine Pooling-Operation angewendet, hier One-Max-Pooling (3). Aus den Ergebnissen, die jeder Filter auf der Dokumentmatrix erzeugt hat, wählt One-Max-Pooling je Filter jeweils das Ergebnis mit der größten Aktivierung aus und verwirft den Rest. Die resultierenden Vektoren enthalten für jeden Filter die jeweils größte Aktivierung in der Dokumentmatrix. Die Vektoren werden daraufhin konkateniert (4).

Eine vollständig verknüpfte Ebene verbindet diesen Vektor mit den Ausgabeneuronen (5). In dieser Ebene erlernt das Modell, Wortmuster mit Ausgabeklassen zu assoziieren. Eine Softmax-Operation wandelt die Ausgaben des Netzwerks in eine Wahrscheinlichkeitsverteilung um.

Überblick: Hyperparameter des CNNs

CNNs zur Klassifikation von Text besitzen eine Reihe von Hyperparametern, die vor dem Training festgelegt werden müssen und die Fähigkeit des Netzwerks, Muster zu erlernen, erheblich beeinflussen. Die Hyperparameter des Word-Embeddings sind nicht Teil des Modells, beeinflussen jedoch ebenfalls die Ergebnisse und werden daher hier aufgelistet:

WORD2VEC: VEKTORGRÖSSE: Die Vektorgröße entscheidet, wie gut das Netzwerk Wörter voneinander unterscheiden kann. Größere Wortvektoren führen zu besserer Unterscheidbarkeit zweier verschiedener Wörter. Durch größere Wortvektoren wird es allerdings ebenfalls unwahrscheinlicher, dass Muster mit ähnlichen Wörtern durch einen einzigen Filter erkannt werden können.

WORD2VEC: WORTHÄUFIGKEIT: Die Worthäufigkeit gibt an, wie häufig ein Wort innerhalb des zum Training des Word-Embeddings verwendeten Korpus auftreten muss, um in das Wörterbuch des Word-Embeddings aufgenommen zu werden. Durch diesen Hyperparameter werden Wörter herausgefiltert, die nur sehr selten im Korpus erscheinen und für die daher keine guten Vektoren erlernt werden können. Auch sinkt dadurch die Größe des Word-Embeddings erheblich. Wird dieser Wert zu hoch gewählt, werden ebenfalls Wörter aus dem Wörterbuch ausgeschlossen, die für eine erfolgreiche Klassifikation relevant sind.

WORD2VEC: TRAINIERBARE VEKTOREN: Die Vektoren des Word-Embeddings werden vor dem Training eines CNNs trainiert und daraufhin nicht modifiziert. Es ist möglich, während der Optimierung der Netzwerkgewichte auch die Word-Embeddings zu optimieren, da durch eine Anpassung der Embeddings auf das konkrete Klassifikationsproblem meist bessere Ergebnisse erzielt werden können.

LÄNGE DER DOKUMENTMATRIX: Obwohl die Netzwerkarchitektur aufgrund des Poolings grundsätzlich mit Dokumenten variabler Länge umgehen kann, muss die Länge der Dokumentmatrix festgelegt werden, damit CNN-Implementierungen entsprechend Speicherplatz reservieren können. Die Länge der Dokumentmatrix sollte so gewählt werden, dass ein Großteil der Dokumente vollständig in die Dokumentmatrix passt. Ist ein Dokument zu kurz, wird es analog zu unbekannten Wörtern um Zufallsdaten oder um einen speziell für unbekannte Wörter verwendeten Vektor ergänzt. Zu lange Dokumente werden abgeschnitten. Wird die Länge der Dokumentmatrix zu klein gewählt, gehen dabei möglicherweise für die Klassifikation relevante Informationen verloren.

FILTERANZAHL: Die Anzahl der Filter legt fest, wie viele unterschiedliche Wortmuster das Modell erlernen kann. Die benötigte Anzahl an Filtern hängt von der Komplexität des Klassifikationsproblems ab. Wird diese Anzahl zu klein gewählt, kann das Netzwerk nicht alle relevanten Muster erlernen. Wird die Anzahl der Filter zu groß gewählt, werden mehrere Filter redundante Informationen enthalten und die Ausführungszeit des Modells sowohl im Training als auch in der Inferenz negativ beeinflussen.

FILTERLÄNGEN: Die Längen der Filter werden analog zur Anzahl der Filter ebenfalls basierend auf er Komplexität des Klassifikationsproblems gewählt. Längere Filter können längere Wortmuster erlernen. Ist dies für ein Klassifikationsproblem nicht notwendig, haben längere Filter keinen Vorteil gegenüber kürzeren Filtern. Filter unterschiedlicher Längen können parallel verwendet werden.

AKTIVIERUNGSFUNKTION CONVOLUTION: Die nach der Convolution-Operation angewandte Aktivierungsfunktion beeinflusst den Lernprozess des Netzwerks. Eine Untersuchung von Zhang und Wallace [119] hat gezeigt, dass Rectified-Linear-Units und der Tangens Hyperbolicus auf mehreren Datensätzen die besten Ergebnisse erzielten.

3 ANFORDERUNGSDOKUMENTE IN DER AUTOMOBILINDUSTRIE

Im Grundlagenkapitel wurden die Techniken eingeführt, die in dieser Arbeit zur Klassifikation der Inhalte von Anforderungsdokumenten in Anforderungen und Informationen eingesetzt werden. Dennoch ist es für die erfolgreiche Lösung eines Datamining-Problems notwendig, einen umfassenden Überblick und ein tiefgehendes Verständnis über die Daten und das zu lösende Problem zu erhalten, bevor mit der Anwendung von Lösungen begonnen wird. Daher wird in diesem Kapitel die Anwendungsdomäne vorgestellt.

Die Klassifikation der Inhalte von Anforderungsdokumenten wurde an Dokumenten aus der Automobilindustrie durchgeführt. Als Basis für die Analyse dienten viele Komponentenlastenhefte und Systemlastenhefte aus den Bereichen Außen- und Innenraumelektronik sowie Fahrerassistenzsysteme und Telematik. Die Arbeit ist daher primär in der Automotive-Domäne angesiedelt. Alle Erläuterungen und Beispiele insbesondere in diesem Kapitel haben einen Bezug zu dieser Domäne. Dennoch ist der vorgestellte Ansatz grundsätzlich auch in anderen Domänen anwendbar. Dies setzt für optimale Ergebnisse allerdings eine erneute Analyse der Anwendungsdomäne voraus.

Dokumenttypen

Es wird zwischen zwei Dokumenttypen unterschieden: *Systemlastenhefte* (SLHs) und *Komponentenlastenhefte* (KLHs). Systemlastenhefte enthalten alle Anforderungen zu einem zusammenhängenden System. Ein *System* in einem Automobil ist eine Sammlung von *Komponenten*, die zusammen eine bestimmte Aufgabe übernehmen. Beispielsweise besteht das System Scheibenwischer unter anderem aus den Komponenten Wischmotor, Lenkstockschalter, Wischwasserpumpe, Steuergerät und Regensensor. Zusammen ermöglichen diese Komponenten dem Fahrer, die Frontscheibe zu wischen und zu waschen. Im Systemlastenheft zu diesem System werden die *Funktionen* dieses Systems spezifiziert. Weiterhin wird spezifiziert, welche Beiträge jede einzelne Komponente zu den Funktionen dieses Systems leistet.

In einem Komponentenlastenheft wird jeweils eine Komponente spezifiziert. Im Gegensatz zu den endbenutzerorientierten Systemlastenheften werden Komponenten sehr viel technischer beschrieben. Ein Komponentenlastenheft enthält beispielsweise Informationen über die Dimensionen der Komponente, die Belegung der einzelnen Pins von Ein-/Ausgabeschnittstellen und Anforderungen an Robustheit gegen Witterung. Weiterhin kann eine Komponente Funktionen mehrerer Systeme umsetzen (das Steuergerät für den Scheibenwischer kann ebenfalls die Steuerung der Beleuchtung übernehmen). Abbil-

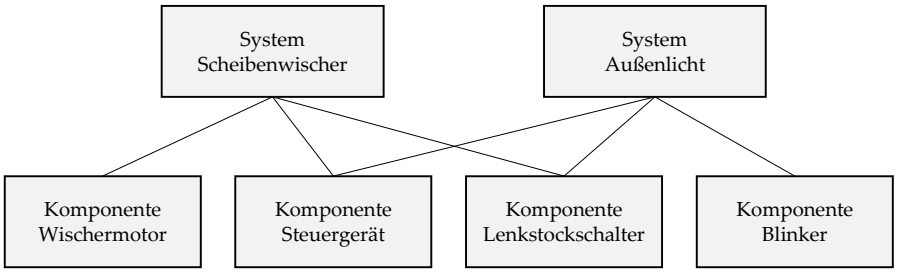


Abbildung 3.1: Beziehungen zwischen Komponenten und Systemen

dung 3.1 veranschaulicht den Zusammenhang zwischen Systemen und Komponenten.

Darüber hinaus werden noch Testspezifikationen für die Lastenhefte erstellt. In diesen werden Testfälle für alle Anforderungen spezifiziert. Diese sind jedoch nicht Fokus dieser Arbeit und werden daher nicht weiter erläutert.

3.1 SYSTEMLASTENHEFTE

In Systemlastenheften werden vollständige Systeme spezifiziert. Zur Organisation der Lastenheftinhalte ist das Lastenheft in die Kapitel Einleitung, Anforderungen an das System, Anforderungen an beteiligte Systeme, Anforderungen an beteiligte Komponenten und Anhang aufgeteilt:

Kapitel in Systemlastenheften

EINLEITUNG: Im Einleitungskapitel werden keine Anforderungen spezifiziert. Dieses Kapitel enthält allgemeine Informationen über das spezifizierte System und Prozessinformationen. Dazu gehören eine kurze, informelle Beschreibung der Funktion des Systems, der Einordnung im Gesamtfahrzeug, Kontaktinformationen der verantwortlichen Personen und Regelungen zu Kommunikation und Änderungsmanagement.

ANFORDERUNGEN AN DAS SYSTEM: In diesem Kapitel werden die Funktionen des Systems spezifiziert. Jede Funktion wird nach einem einheitlichen Schema spezifiziert. Dieses Schema umfasst die Punkte Vorbedingungen, Voraussetzungen, Auslöser, Ausführung und Ende-/Abbruchbedingungen. Eine Funktion kann nur ausgeführt werden, wenn Vorbedingungen und Voraussetzungen erfüllt sind. Die Funktion wird gemäß Ausführung ausgeführt, wenn die Auslöser eintreten und (vorzeitig) beendet, wenn die Ende-/Abbruchbedingungen eintreten. Dank dieses einheitlichen Schemas ist es möglich, die Anforderungen in der Funktionsspezifikation sehr kurz zu halten. Weiterhin werden in diesem Kapitel die Anwendungsfälle des Systems und Anforderungen an die funktionale Sicherheit spezifiziert.

ANFORDERUNGEN AN BETEILIGTE SYSTEME: In diesem Kapitel werden die Abhängigkeiten zu anderen Systemen spezifiziert. Die Abhängigkeiten werden als Funktionsbeiträge spezifiziert und nach System in Unterkapitel gruppiert. In einem Funktionsbeitrag werden die Funktionen des beteiligten Systems durch ergänzende Anforderungen erweitert. Dies umfasst meist gemeinsam genutzte Ressourcen und Ereignisse, die beide Systeme betreffen.

ANFORDERUNGEN AN BETEILIGTE KOMPONENTEN: Analog zum Kapitel beteiligte Systeme werden in diesem Kapitel Anforderungen an die Komponenten spezifiziert, welche die Funktionen des Systems umsetzen. Dies geschieht mittels Komponentenbeiträgen. In jedem Komponentenbeitrag werden Anforderungen spezifiziert, welche von der adressierten Komponente umzusetzen sind. Ebenfalls werden diese genau wie Funktionsbeiträge nach Komponente in Unterkapitel gruppiert.

ANHANG: Im Anhang befinden sich meist Verweise auf weitere, relevante Dokumente und ein Glossar.

3.2 KOMPONENTENLASTENHEFTE

Komponentenlastenhefte sind im Gegensatz zu Systemlastenheften wesentlich technischer spezifiziert und enthalten Spezifikationen zu beispielsweise den Ein- und Ausgabeschnittstellen, physikalischen Dimensionen und zur Montage im Gesamtsystem. Weiterhin werden viele für den Fertigungsprozess wichtige Informationen im Komponentenlastenheft festgehalten. Auch Komponentenlastenhefte sind in einheitlich in die folgenden Kapitel unterteilt:

Kapitel in Komponentenlastenheften

EINLEITUNG: In diesem Kapitel befinden sich allgemeine Informationen zur Handhabung des vorliegendes Komponentenlastenhefts. Diese sind nicht komponentenspezifisch.

ALLGEMEINE VORGABEN: Allgemeine Vorgaben umfassen eine Kurzbeschreibung der Komponente, Identifikationsnummern, Anforderungen an die zu verwendenden Entwicklungsprozesse und Termine.

ANSPRECHPARTNER: Genau wie auch in Systemlastenheften werden hier die Personen aufgelistet, die für die Entwicklung der jeweiligen Komponente verantwortlich sind.

DOKUMENTATION: In diesem Kapitel wird definiert, in welchem Umfang die Komponente vom Auftragnehmer dokumentiert werden muss und was bei der Dokumentation beachtet werden muss.

KOMPONENTENUMGEBUNG: Hier wird spezifiziert, wo die Komponente verbaut wird und unter welchen Bedingungen (Temperatur, Witterung) sie arbeiten muss.

EIGENSCHAFTEN DER KOMPONENTE: Dieses Kapitel ist das größte Kapitel im Komponentenlastenheft. Hier werden die Funktionen der Komponente spezifiziert. Jede Funktion wird ähnlich wie auch im Systemlastenheft, nur näher an der Hardware spezifiziert. Weiterhin werden in diesem Kapitel Eigenschaften wie Schnittstellen und Geometrie spezifiziert.

ANHANG: Wie auch das Systemlastenheft enthält der Anhang Verweise auf andere relevante Dokumente und ein Glossar.

3.3 VORLAGEN

Die Spezifikation von Systemen und Komponenten erfolgt immer nach dem gleichen Muster. Damit System- und Komponentenlastenhefte einheitlich sind, werden diese nicht von Grund auf neu erstellt, sondern von Vorlagen abgeleitet. Dies hat zur Folge, dass alle Lastenhefte eines Typs identische Strukturen haben. Dies erhöht die Lesbarkeit der Dokumente, da bestimmte Inhalte immer an derselben Position im Dokument gefunden werden können.

In den Vorlagen werden die Kapitel und system- bzw. komponentenübergreifende Inhalte definiert. Insbesondere in Komponentenlastenheften werden die meisten Lastenheftinhalte (ausgenommen das Kapitel Eigenschaften der Komponente) bereits größtenteils in der Vorlage definiert, da diese für alle Komponenten identisch sind. An Stellen, die system- oder komponentenspezifisch sind, werden Platzhalter verwendet. Für größere spezifische Inhalte wie Funktionen oder Komponentenbeiträge werden Schablonen zur Verfügung gestellt.

3.4 STRUKTUR

Die Struktur, in denen Lastenhefte erstellt werden, muss gewisse Kriterien erfüllen, damit die Lastenhefte für die verschiedenen Arbeitsschritte in Anforderungsmanagementprozessen geeignet sind. Beispielsweise müssen einzelne Anforderungen eines Lastenhefts eindeutig identifizierbar und aus anderen Dokumenten referenzierbar sein, damit beispielsweise ein Testfall mit der dazugehörigen Anforderung zwecks Nachverfolgbarkeit verknüpft werden kann. Weiterhin ist es in den Prozessen von Vorteil, bestimmte Teile eines Lastenhefts mit zusätzlichen Informationen zu versehen. Daher sind Lastenhefte während der Bearbeitung ähnlich wie relationale Datenbanken aufgebaut

und werden erst nach der Finalisierung zur Ablage in Dokumentform überführt.

Objekte

Ein Lastenheft besteht aus vielen *Objekten*. Objekte werden benutzt, um den Inhalt des Lastenhefts festzuhalten. Jedes Objekt enthält einen möglichst atomaren Bestandteil eines Lastenhefts, beispielsweise eine Überschrift, eine Anforderung, eine Abbildung oder einen Stichpunkt einer Aufzählung.

Attribute

Objekte können beliebig viele *Attribute* besitzen. Jedes Attribut kann ähnlich wie Spalten in relationalen Datenbanken eine bestimmte Art von Information speichern. Einige der Attribute sind fest vorgegeben, weitere können nach Notwendigkeit hinzugefügt werden. Die in den analysierten Dokumenten am häufigsten vorkommenden und für diese Arbeit relevanten Attribute sind in Tabelle 3.1 aufgelistet.

Weiterhin können alle Objekte jeweils ein Elternobjekt und beliebig viele Kinderobjekte besitzen. Alle Kinderobjekte eines Objekts haben eine feste Ordnung. Somit ist es möglich, die Objekte in einer baumartigen Struktur zu organisieren und eine aus Kapiteln und Unterkapiteln bestehende Struktur zu erstellen.

Trace-Links

Ebenfalls können Objekte mit Trace-Links miteinander unidirektional verknüpft werden. Dies ist sowohl zwischen zwei Objekten desselben Lastenhefts als auch zwischen zwei Objekten unterschiedlicher Dokumente möglich. Mit Trace-Links können beispielsweise die Relationen zwischen Testfällen und Anforderungen und zwischen High-Level-Anforderungen und Low-Level-Anforderungen dokumentiert werden.

3.5 INHALTE

Zur Dokumentation von Anforderungen können unterschiedliche Formate genutzt werden. In den untersuchten Lastenheften wurde fast ausschließlich natürliche Sprache (Deutsch und Englisch) verwendet. Es ist ebenfalls möglich, Anforderungen mit graphischen Hilfsmitteln (z.B. Aktivitätsdiagramme [15]), formalen Notationen oder Boiler-Plates (siehe dazu auch Abschnitt 8.1.1) zu dokumentieren.

Objekttyp Anforderung

Der Großteil der Lastenhefte besteht aus natürlichsprachlichen Anforderungen. Jede einzelne Anforderung wird in einem separaten Objekt abgelegt. Anforderungen werden mit dem Objekttyp *Anforderung* gekennzeichnet. Anforderungen zeichnen sich oft durch spezielle Formulierungen aus, die besonders zur genauen Beschreibung von Systemverhalten geeignet sind. Dazu zählen beispielsweise „Das System muss ...“, „Während ..., wird ...“ und „Wenn ..., dann ...“. Objekte mit diesen Formulierungen sind allerdings nicht notwendigerweise immer Anforderungen. In der folgenden Aufzählung werden einige Beispiele für typische Anforderungen gegeben.

ATTRIBUT	BESCHREIBUNG
ID	Eine beliebige Zeichenkette, die das Objekte im Lastenheft eindeutig identifiziert.
Inhalt	Inhalt des Objekts. Neben Text können hier auch Abbildungen oder andere, beliebige Dateiformate eingebettet werden. Leer, wenn das Objekt eine Überschrift ist.
Überschrift	Überschrift des Objekts. Kann nur Text enthalten. Leer, wenn das Objekt keine Überschrift ist.
Objekt-Level	Hierarchische Tiefe des Objekts. Objekte auf oberster Ebene haben das Level 1, jedes Objekt darunter Level 2, usw.
Objektnummer	Fortlaufende Nummer des Objektes. In jeder Hierarchieebene hat das erste Objekt die Nummer 1. Vergleichbar mit Kapitelnummern.
Objekttyp	Der Typ des Inhalts des Objekts. Kann frei gewählt werden. Üblich sind unter anderem *Anforderung*, *Information* und *Überschrift*.
Editiertyp	Bei Objekten, die aus einer Vorlage übernommen wurden, gibt dieses Attribut an, ob und wie der Inhalt dieses Objekts geändert werden muss.
Reifegrad	Gibt den Reifegrad des Objekts an. Wird vorrangig für Anforderungen verwendet. Objekte starten mit dem Reifegrad *offen* und bekommen i.d.R. den Reifegrad *abgestimmt*, nachdem sie einem Review sowohl vom Auftraggeber als auch vom Auftragnehmer unterzogen wurden.
PVM	Empfohlene Verifikationsmethode (engl. „Potential Verification Method“). Eine Abschätzung eines RE-Experten, wie die Anforderung später am besten zu verifizieren ist (z.B. Hardware-in-the-Loop, Simulation, Review). Muss nicht mit der tatsächlich gewählten Verifikationsmethode übereinstimmen. Mehr in Kapitel 7.

Tabelle 3.1: Übersicht Objektattribute in Lastenheften

Objekttyp Information

Neben Anforderungen enthalten die Lastenhefte weitere Zusatzinformationen. Zusatzinformationen ergänzen die Anforderungen, verbessern das Verständnis des Lastenhefts und reduzieren damit den Kommunikationsbedarf zwischen den beteiligten Personen. Zu Zusatzinformationen gehören Erklärungen für Anforderungen, Beispiele, Begründungen, allgemeine Systembeschreibungen, Informationen zum Inhalt bestimmter Kapitel, veranschaulichende Abbildungen, Verweise auf andere relevante Dokumente und sonstige Anmerkungen. Auch die Auflistung von Ansprechpartnern gehört zu den Zusatzinformationen. Objekte mit Zusatzinformationen werden mit dem Objekttyp *Information* gekennzeichnet. Diese sind keine Anforderungen und tragen damit nicht zur Funktion des Systems bei; wenn alle Zusatzinformationen aus einem Lastenheft entfernt werden, hat dies keine Auswirkungen auf das zu entwickelnde System. Im Gegensatz zu Anforderungen lassen sich Zusatzinformationen meist nicht an wenigen Formulierungen erkennen, sondern zeichnen sich vielmehr durch bestimmte, teils auch für das jeweilige Lastenheft spezifische Wörter aus. Die folgende Aufzählung enthält Beispiele häufig vorkommender Zusatzinformationen.

- Das vorliegende Dokument beschreibt ein komponentenübergreifendes System.
- In der Systembeschreibung Wischen/Waschen sind sämtliche Wisch- und Waschfunktionen beschrieben.
- Im folgenden Abschnitt sind die Ansprechpartner für das System und die beteiligten Komponenten aufgeführt.
- Prinzip Darstellung: Siehe Objekt [OBJ-1234].
- Anmerkung: Bei Klemme 1 sind keine gültigen Getriebestellungen verfügbar, es wird konstant „N“ gesendet. Damit hat Klemme 1 keine Auswirkungen auf die Funktion.
- Erklärung: Bei einem hängenden Taster kann der Fahrer den Hebel auf Stellung 0 bringen, dann wird dort der hängende Taster erkannt.

Weitere Objekttypen

Lastenhefte enthalten neben Anforderungen und Zusatzinformationen noch weitere Inhalte, die durch weitere Objekttypen gekennzeichnet werden. Überschriften werden mit dem Typ *Überschrift* gekennzeichnet. Für Definitionen, dir für das gesamte Dokument gelten, wird der Typ *Vordefinition* verwendet. In besonderen Situationen werden weitere Typen verwendet.

Strukturen

Während der Großteil der Lastenhefte in natürlichsprachlichen Sätzen formuliert ist, besteht ein nicht unwesentlicher Teil aus anderen Strukturen. Je nach Situation sind diese besser zur Dokumentation

der Inhalte geeignet. Die am häufigsten vorkommenden Strukturen werden im Folgenden vorgestellt.

Aufzählungen

Wenn viele gleichartige Eigenschaften eines Systems spezifiziert werden, bieten sich *Aufzählungen* an. Für jedes Element der Aufzählung wird dabei ein eigenes Objekt verwendet, welche alle unter einem gemeinsamen Elternobjekt gesammelt werden:

> Reaktionswischen muss bei folgenden Aktionen ausgeführt werden:
>
> Umschalten von Stufe 0 auf Intervall 1
>
> Umschalten von Intervall 1 auf Intervall 2
>
> Umschalten von Stufe 0 auf Intervall 2

Aufzählungen werden auf häufig in der Spezifikation von Funktionen in Systemlastenheften verwendet:

> 2.6.2 FUNKTION FRONTSCHEIBE WISCHEN
>
> ...
>
> 2.6.2.3 AUSLÖSER
>
> Regensensor erkennt Regen
>
> Hebel auf Position 1 gestellt
>
> Hebel auf Position 2 gestellt

Für jedes Objekt der Aufzählung wird separat der Objekttyp festgelegt. Jedes Objekt kann somit eine Anforderung sein, obwohl sich die grammatikalischen Strukturen deutlich von anderen Anforderungen unterscheiden. Dies ermöglicht es zudem, auch in Aufzählungen Zusatzinformationen hinzuzufügen.

Abbildungen

Abbildungen können verwendet werden, um komplizierte Zusammenhänge einfach darzustellen. In Lastenheften wird dies in vielen unterschiedlichen Situationen genutzt. Blockschaltbilder zeigen den Zusammenhang zwischen verschiedenen Teilen eines größeren Systems, beispielsweise die Vernetzung von mehreren Komponenten mit einem Bussystem und die Interaktion von Systemen mit benachbarten Systemen. Aktivitätsdiagramme werden genutzt, um den Ablauf komplizierter Funktionen zu veranschaulichen [15]. Andere Abbildungstypen sind möglich.

Ob eine Abbildung eine Anforderung oder eine Zusatzinformation ist, hängt vom Inhalt und vom Kontext der Abbildung ab. Ist die Abbildung die einzige Dokumentation eines Details des spezifizierten Systems, ist es eine Anforderung. Dient die Abbildung nur zur Unterstützung des Textes, ist sie eine Zusatzinformation.

Definitionen

In Lastenheften werden oft für bestimme Parameter des Systems konkrete Werte festgelegt. Anstatt bei jeder Definition eines Parameters eine vollständige Anforderung zu formulieren („Die Abmessungen

der Komponente müssen 100 mm x 50 mm x 20 mm betragen."), wird dies in den Lastenheften häufig als *Definition* abgekürzt:

- Abmessungen L x B x H: 100 mm x 50 mm x 20 mm
- Spannung: 12 V
- Dauer: 10 ms

Formeln

Mathematische *Formeln* werden in Lastenheften genutzt, wenn Zusammenhänge zwischen mehreren Variablen spezifiziert werden. Natürliche Sprache kann ebenfalls zur Spezifikation dieser Zusammenhänge genutzt werden. Allerdings ist es mit natürlicher Sprache oft komplizierter, diese Zusammenhänge eindeutig und korrekt zu spezifizieren.

Tabellen

Wenn mehrere Eigenschaften vieler Elemente spezifiziert werden müssen, werden dafür oft *Tabellen* benutzt. Wie auch bei sämtlichen anderen Strukturen entscheidet der Inhalt der Tabelle über den Objekttyp.

3.6 REVIEW VON SPEZIFIKATIONEN: AKTUELLER STAND

Bei jedem Lastenheft muss sichergestellt werden, dass dieses bestimmte Qualitätskriterien erfüllt, damit in Folgeprozessen keine Probleme oder Verzögerungen entstehen. Erfüllt ein Lastenheft diese Qualitätskriterien nicht, kann es passieren, dass das entwickelte System bzw. die entwickelte Komponente nicht den Erwartungen des Auftraggebers entspricht. Dies führt zu erheblich höheren Kosten, da das nachträgliche Ändern von Anforderungen umfangreiche Auswirkungen auf sämtliche nachfolgende Prozesse hat. Um dies zu verhindern, werden viele unterschiedliche Aspekte des Lastenhefts geprüft.

1. Es werden verschiedene formale Prüfungen durchgeführt. Dabei wird unter anderem untersucht, ob alle Platzhalter der Vorlage befüllt wurden, ob alle in der Vorlage vorgegebenen Inhalte vorhanden sind und ob bei bestimmten Attributen aller Objekte gültige Werte gesetzt sind. Diese Überprüfung ist rein formal und stellt nicht sicher, dass die Inhalte tatsächlich sinnvoll sind.
2. Eine Rechtschreib- und Grammatikprüfung erhöht die Lesbarkeit des Dokuments.
3. Anforderungen werden inhaltlich überprüft. Dabei wird insbesondere auf die in Abschnitt 2.1.2 erwähnten Qualitätsmerkmale von Anforderungen geachtet: Vollständigkeit, Eindeutigkeit,

Konsistenz, Notwendigkeit, Atomarität, Lösungsneutralität, Umsetzbarkeit, Nachverfolgbarkeit und Testbarkeit. Diese Überprüfungen sind von besonderer Wichtigkeit, da die Verletzung dieser Regeln Auswirkungen auf die Bedeutung und Auslegbarkeit der Anforderungen hat und damit potenziell zu unerwünschten Eigenschaften des finalen Produkts führen kann.

4. Alle Attribute werden inhaltlich überprüft. Dazu zählt unter anderen der Objekttyp und damit die saubere Trennung von Anforderungen und Informationen. Andere Attribute wie PVM werden ebenfalls untersucht.

Grenzen aktueller Werkzeuge

Für die Überprüfung dieser Attribute gibt es rudimentäre Werkzeugunterstützung. Mit Werkzeugen ist es möglich, sämtliche formalen Tests durchzuführen. Findet das Werkzeug eine Regelverletzung, wird der Benutzer durch eine Warnung auf den Fehler aufmerksam gemacht und zur Änderung aufgefordert. Mit Werkzeugen ist des derzeit nicht möglich zu überprüfen, ob beispielsweise die Befüllung der Platzhalter sinnvoll ist. Dies muss manuell überprüft werden. Nur für die einfachsten Aufgaben werden derzeit Werkzeuge verwendet.

Insbesondere für die aufwändigen und für Folgeprozesse deutlich relevanteren Prüfungen gibt es keine automatische Lösung, da hierfür natürliche Sprache verarbeitet werden muss und es für die Überprüfung der verschiedenen Qualitätskriterien notwendig ist, komplexe Zusammenhänge zwischen den Anforderungen auszuwerten. Daher werden Anforderungsdokumente größtenteils manuell überprüft, wobei jedes einzelne Objekt untersucht wird. Ebenfalls wird dabei untersucht, ob der Objekttyp der Objekte korrekt ist. Dieser Prozess ist aufgrund der Größe der Anforderungsdokumente zeitaufwändig und fehleranfällig.

Der Einsatz von Werkzeugen für diese Tests wird weiterhin dadurch erschwert, dass Ansätze, die auf natürlicher Sprache arbeiten, aufgrund deren probabilistischer Natur nicht alle Fehler finden können. Ein Werkzeug, welches nur einen Teil der Fehler findet, ist nutzlos, wenn nach dem Einsatz des Werkzeugs weiterhin das gesamte Dokument auf die verbleibenden Fehler untersucht werden muss [19]. Entwickler verlassen sich in der Regel nicht auf ein Werkzeug, wenn dieses in bestimmten Situationen nicht funktioniert.

Allerdings werden beispielsweise auch Menschen das Problem der Klassifikation von Objekten in Anforderungen und Informationen nicht perfekt lösen. Solange das Werkzeug besser ist, kann durch den Werkzeugeinsatz Zeit und damit Kosten gespart werden [20]. Der Großteil der Arbeit beschäftigt sich daher am Beispiel des Klassifikationsproblems Information/Anforderung mit der Entwicklung von einem Werkzeug, mit welchem trotz teilweise fehlerhafter Ausgaben signifikante Verbesserungen im Reviewprozess erzielt werden können.

4 KLASSIFIKATION VON ANFORDERUNGEN UND INFORMATIONEN

Um ein Werkzeug zur Unterstützung der Lastenheftentwickler bei der Klassifikation von Objekten in Anforderungen und Informationen zu konstruieren, muss zuerst ein Mechanismus entwickelt werden, der diese Klassifikation selbstständig und zuverlässig durchführen kann. In diesem Kapitel wird beschrieben, wie unter Verwendung von Standardprozessen und unter Zuhilfenahme der in Kapitel 3 vorgestellten Informationen über Lastenhefte ein auf Convolutional-Neural-Networks basierendes Klassifikationsmodell konstruiert wird, der Informationen von Anforderungen unterscheiden kann.

4.1 DER KDD-STANDARDPROZESS

Im *Data Mining and Knowledge Discovery Handbook* [75] beschreiben Maimon und Rokach einen allgemeinen und in vielen Gebieten anwendbaren Prozess zur Bewältigung von Data-Mining-Aufgaben auf beliebigen Datensätzen. Der „Knowledge Discovery in Databases Process“, kurz KDD-Prozess, wird auch in dieser Arbeit verwendet und daher kurz vorgestellt.

Der Prozess unterteilt sich in die folgenden neun Schritte. Der gesamte Prozess ist iterativ, d.h. es ist häufig notwendig, zu bereits erledigten Schritten zurückzugehen und Anpassungen vorzunehmen. Der Prozess enthält zudem keine exakten Anweisungen, da die jeweils notwendigen Schritte und Techniken stark vom Anwendungsgebiet und von den zu erreichenden Zielen abhängen.

Schritte des KDD-Prozesses

1. VERSTEHEN DER ANWENDUNGSDOMÄNE. Bevor mit der eigentlichen Arbeit begonnen werden kann, muss die Anwendungsdomäne untersucht werden. In diesem Schritt werden die Ziele des KDD-Projekts festgelegt und die Datenbasis untersucht. Insbesondere wird die Datenbasis auch auf Anomalien und andere Besonderheiten untersucht, die spätere Schritte ggf. erschweren können.

2. AUSWÄHLEN VON DATEN UND ERSTELLEN EINES DATENSATZES. Als nächstes werden die zur Verfügung stehenden Daten gesichtet und in einem Datensatz gesammelt. Dabei muss insbesondere ermittelt werden, welche Attribute der Daten in den Datensatz

übernommen werden. Dieser Schritt muss sorgfältig durchgeführt werden, da der Erfolg des KDD-Projekts von der Qualität des Datensatzes abhängt.

3. VORVERARBEITUNG UND DATENBEREINIGUNG. Im nächsten Schritt wird die Zuverlässigkeit der Daten verbessert. Dabei wird vor allem auf fehlende und fehlerhafte Daten, Rauschen und Ausreißer geachtet. Es werden geeignete Maßnahmen unternommen, um diese Probleme zu beseitigen. Fehlerhafte Daten können korrigiert, fehlende Daten ergänzt und Ausreißer entfernt werden.

4. DATENTRANSFORMATION. Der Datensatz wird nun in ein Format transformiert, das für die Anwendung von Data-Mining-Algorithmen besser geeignet ist. Techniken, die dabei zur Anwendung kommen sind Feature-Auswahl und Feature-Reduktion.

5. AUSWAHL DER GEEIGNETEN DATA-MINING-AUFGABE. Nach diesen vorbereitenden Schritten kann auf Basis der gewonnenen Erkenntnisse und gesetzten Ziele eine geeignete Data-Mining-Aufgabe gewählt werden. Typische Aufgaben sind beispielsweise Klassifikation, Regression und Clustering.

6. AUSWAHL EINES DATA-MINING-ALGORITHMUS. Passend zur gewählten Aufgabe wird nun ein Data-Mining-Algorithmus bestimmt. Jeder Algorithmus (neuronale Netzwerke, Naive Bayes, k-NN, Support Vector Machines, Entscheidungsbäume...) hat gewisse Vor- und Nachteile. Der Algorithmus muss daher auch passend zur Beschaffenheit der Daten ausgewählt werden.

7. ANWENDEN DES DATA-MINING-ALGORITHMUS. Nun wird eine Implementierung mit dem gewählten Algorithmus erstellt und auf den Datensatz angewendet. Da die meisten Algorithmen gewisse Einstellmöglichkeiten bieten, wird dies solange mit unterschiedlichen Hyperparametern wiederholt, bis ein zufriedenstellendes Ergebnis erreicht ist.

8. AUSWERTUNG DER ERGEBNISSE. Nach jeder Anwendung des Algorithmus werden die Ergebnisse ausgewertet. Insbesondere wird untersucht, ob die vorher gesetzten Ziele erreicht wurden. Dazu können unterschiedliche Methoden wie statistische Auswertungen mit gängigen Metriken, stichprobenartige qualitative Auswertungen und empirische Studien mit Anwendern eingesetzt werden.

9. ANWENDEN DER GEWONNENEN ERKENNTNISSE. Nachdem die Auswertungen das Erreichen der Ziele bestätigt haben, können die gewonnenen Erkenntnisse und Modelle in ein System integriert werden. In diesem Schritt zeigt sich, wie gut und gewissenhaft die vorangegangenen Schritte durchgeführt wurden,

da das System nun erstmals außerhalb von Laborbedingungen arbeitet. Dies bedeutet auch, dass gegebenenfalls Anpassungen notwendig werden, wenn sich die Beschaffenheit der Daten in Zukunft ändert.

Die Ergebnisse des ersten Schrittes wurden bereits in Kapitel 3 erläutert. Das Vorgehen für die weiteren Schritte zur Konstruktion eines auf Convolutional-Neural-Networks basierenden Klassifikators wird in den folgenden Abschnitten und Kapiteln beschrieben. Die Leistung dieses Klassifikators wird mit weiteren, in der Textklassifikation üblichen Klassifikationsmodellen verglichen. Die dafür zusätzlich notwendigen Vorbereitungen werden ebenfalls in den folgenden Abschnitten beschrieben.

4.2 DATENSAMMLUNG UND -AUSWAHL

Zuerst wurden die in einer IBM DOORS-Datenbank[1] zur Verfügung stehenden Daten gesichtet und gesammelt. In dieser Datenbank befinden sich alle derzeit bearbeiteten Anforderungsdokumente. Für diese Arbeit wurden Daten aus den Bereichen Fahrzeuginnen- und Außenraumelektronik, Fahrerassistenzsysteme und Telematik verwendet.

Überblick über die zur Verfügung stehenden Daten

Nach der Extraktion der Daten standen insgesamt 467 vollständige Systemlastenhefte, 317 vollständige Komponentenlastenhefte, 423 Testspezifikationen und 7.499 weitere Dokumente zur Verfügung. Unter den weiteren Dokumenten befinden sich viele Dokumente, die für Verwaltungsaufgaben erstellt wurden oder Metainformationen enthalten. Allerdings befinden sich unter diesen Dokumenten aufgrund der Art und Weise, wie Anforderungsdokumente in der Datenbank abgelegt sind, auch viele Anforderungsdokumente, die nicht als vollständiges Dokument erkannt wurden. Die Systemlastenhefte umfassen durchschnittlich 513 Objekte, die Komponentenlastenhefte sind durchschnittlich 1.117 Objekte groß. Insgesamt standen 8.706 Dokumente mit insgesamt 2.895.160 Objekten und 60.614.990 Wörtern zur Verfügung. 4.420 der Dokumente sind in Deutsch geschrieben, 4.286 der Dokumente in Englisch. Die Sprache eines Dokuments wurde mithilfe stochastischer Verfahren automatisch bestimmt. Diese Ansätze haben zwar Schwierigkeiten mit der Bestimmung der Sprache bei Sätzen mit vielen Fach- und Fremdwörtern, funktionieren für vollständige Dokumente einwandfrei, wenn diese überwiegend Wörter einer Sprache enthalten.

Abbildung 4.1 zeigt die durchschnittliche Länge von Anforderungen und Informationen in der Datenmenge. Ein Großteil der Objekte ist nicht mehr als 40 Wörter lang, nur wenige Objekte sind umfangreicher.

1 Dynamic Object Oriented Requirements System; Anforderungsmanagementsystem

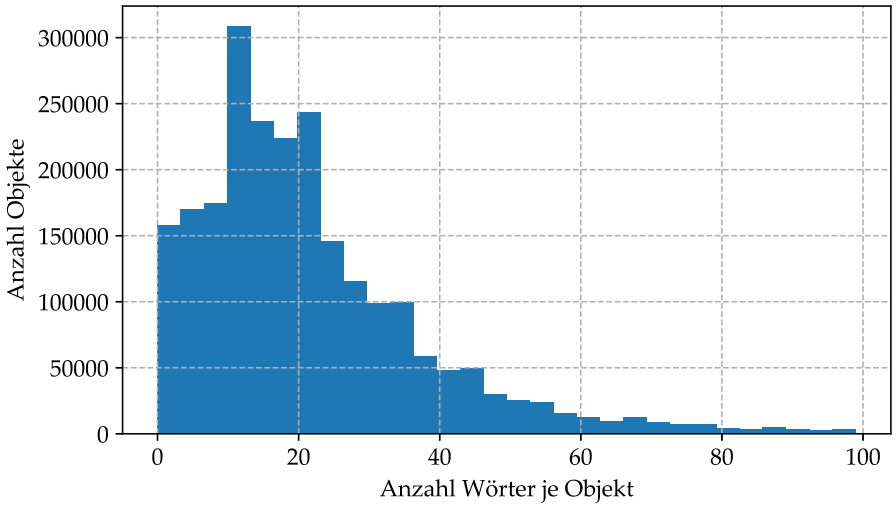


Abbildung 4.1: Histogramm über die Wortanzahl in Spezifikationsobjekten

Diese Daten sind für das später folgende Training eines Klassifikators relevant.

In den Objekten der gesammelten Dokumente wurde bei 90,2% der Objekttyp gesetzt. Insgesamt gibt es 54,3% Anforderungen, 10,8% Informationen, 18,1% Überschriften und 7,0% andere Objekte. Das später für die Datensatzerstellung benutzte Attribut Reifegrad wurde bei 8,1% aller Anforderungen auf *abgestimmt* gesetzt. Bei den restlichen Objekten wurden andere Werte benutzt, die jeweils einen bestimmten Fortschritt des Reviewprozesses signalisieren.

Auswahl von Dokumenten und Objekten für den Datensatz

Aus dieser Datenmenge wurden Objekte nach den folgenden Kriterien für den Datensatz ausgewählt:

1. Das Dokument muss eine Komponente, ein System oder Teile einer Komponente oder eines Systems beschreiben. Die meisten Komponenten und Systeme sind in einem einzigen Dokument beschrieben, es gibt jedoch auch Lastenhefte, deren Inhalte auf mehrere Dokumente aufgeteilt sind. Andere Dokumente wurden ignoriert. Die Filterung nach Dokumenttyp wurde automatisch auf Basis von Metadaten der Dokumente (Dateiname, Attribute) durchgeführt.

2. Die Inhalte des Datensatzes müssen in derselben Sprache geschrieben sein. Da sowohl sehr viele deutschsprachige als auch englischsprachige Dokumente vorhanden sind, wurden zwei Datensätze erstellt, und die folgenden Schritte jeweils sowohl auf dem deutschen Datensatz als auch auf dem englischen Datensatz durchgeführt. Dies ermöglicht es insbesondere in der Auswertung, die Leistung des Klassifikators auf zwei unterschiedlichen Datensätzen zu testen. Zur Sortierung der Objekte nach Sprache wurde die Sprache für jedes Objekt eines Dokuments stochastisch bestimmt. Objekte, deren Sprache nicht als Deutsch oder

Englisch bestimmt wurde, oder bei denen die Zuverlässigkeit der vorhergesagten Sprache unter 90% lag, wurden ignoriert.

3. Der Objekttyp muss entweder auf *Anforderung* oder *Information* gesetzt sein. Andere Objekte wurden verworfen. Dadurch werden ebenfalls Testspezifikationen aussortiert, da für die Objekte einer Testspezifikation andere Objekttypen verwendet werden.

4.3 VORVERARBEITUNG

Zur Verbesserung der Qualität des Datensatzes wurden verschiedene Schritte angewendet, die im Folgenden beschrieben werden.

Vorlagenfilterung

Alle Dokumente wurden auf Basis einer Vorlage erstellt. Diese gibt nicht nur die Struktur und allgemeine Inhalte vor, sondern legt auch die Objekttyp-Klassifikation jedes Objekts der Vorlage fest. Es ist nicht notwendig, dass der Klassifikator diese Objekte richtig klassifizieren kann. Daher wurden Objekte, die aus einer Vorlage übernommen wurden, aus dem Datensatz entfernt. Ob ein Objekt aus einer Vorlage stammt, lässt sich über das Attribut *Identifier* bestimmen.

Duplikate

Der Datensatz enthält viele Duplikate. Ein Objekt ist ein Duplikat eines anderen Objekts, wenn deren Objekttext übereinstimmt. Duplikate wirken sich negativ auf die Validität der Evaluation aus. Ein Objekt, welches sich sowohl im Trainingsdatensatz als auch im Testdatensatz befindet, wird höchstwahrscheinlich richtig klassifiziert und lässt die Ergebnisse positiver erscheinen, als sie tatsächlich sind. Ein Großteil der Duplikate wurde durch die Entfernung von Vorlagenobjekten bereits beseitigt. Dennoch enthält die Datenmenge viele Duplikate, da oftmals Objekte zwischen Lastenheften kopiert werden und ein Lastenheft in verschiedenen Varianten mit marginalen Unterschieden existieren kann. Alle Duplikate wurden entfernt.

Reifegradfilterung

Die Qualität der Lastenhefte in der Datenmenge schwankt stark. Dies betrifft sowohl Objekttext als auch andere Attribute. Anforderungen sind teilweise in Umgangssprache geschrieben und Attribute (insb. auch der Objekttyp) sind teilweise falsch gesetzt. Bei der Anwendung von Data-Mining-Algorithmen wird dies zu unvorhersehbaren Auswirkungen sowohl in der Evaluation als auch im späteren Einsatz führen. Daher werden die Objekte mithilfe des Objektattributs Reifegrad gefiltert. Der Wert *abgestimmt* dieses Attributs gibt an, dass das betroffene Objekt bereits durch einen RE-Experten geprüft wurde. In diesen Objekten wurden bei einer stichprobenartigen Untersuchung weniger Fehler in der Klassifikation des Objekttyps gefunden. Daher wurden nur Objekte in den Datensatz übernommen, die als *abgestimmt* gekennzeichnet sind.

Cluster-Korrektur

Auch nach dieser Filterung enthält die Datenmenge weiterhin einige Objekte, die falsch klassifiziert sind. Eine manuelle Überprüfung der Klassifikation aller Objekte ist aufgrund der großen Anzahl an

Objekten nicht möglich, daher wurde zur weiteren Verbesserung der Datenqualität eine automatische Fehlerkorrektur auf Basis von Clustering durchgeführt.

In der Datenmenge existieren viele Objekte, die nicht identisch, aber sehr ähnlich sind. Ein Beispiel sind die folgenden Anforderungsobjekte, in denen die Schnittstellen von Hardwarekomponenten spezifiziert werden:

> Pin-Nr: 1
> Pin-Bezeichnung: Wischermotor
> Nenn-Spannung: 12 V
> Nenn-Strom: 1 A
> Max-Strom: 4 A
>
> Pin-Nr: 2
> Pin-Bezeichnung: Regensensor
> Nenn-Spannung: 12 V
> Nenn-Strom: 10 mA
> Max-Strom: 20 mA

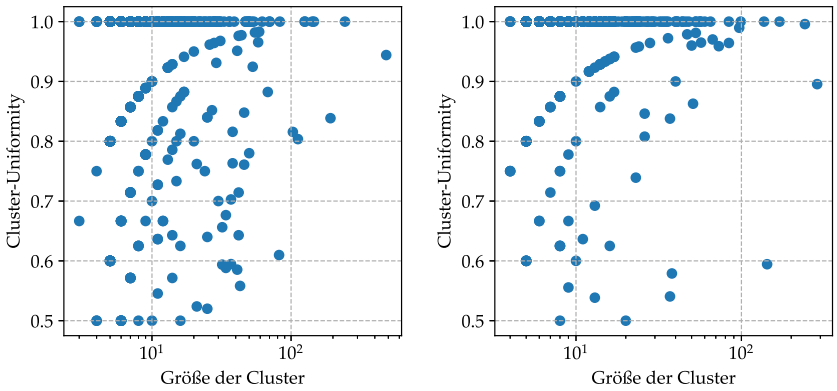
Diese Objekte sind ähnlich, da große Teile des Textes übereinstimmen. Die Klassifikation aller ähnlichen Objekte sollte identisch sein, es befinden sich jedoch Objekte in der Datenmenge, bei denen dies nicht der Fall ist. Daher wird bei ausgewählten ähnlichen Objekten die Klassifikation aneinander angeglichen.

Die Objekte werden zunächst in Cluster ähnlicher Objekte aufgeteilt. Dafür wurde ein Single-Link-Clustering-Algorithmus [4] verwendet. Dieses hierarchisches Cluster-Verfahren setzt im Gegensatz zu partitionierenden Cluster-Verfahren wie k-Means keine vorher festgelegte Cluster-Anzahl voraus. Das ist entscheidend, da die Anzahl der Cluster nicht bekannt ist. Der Single-Link-Clustering-Algorithmus benötigt weiterhin eine Funktion zur Bestimmung der Entfernung zweier Objekte. Dafür wurde ein Textähnlichkeitsmaß, die Levenshtein-Distanz, verwendet.

Die so entstandenen Cluster wurden daraufhin mit der folgenden Metrik untersucht:

$$\mathtt{ClusterUniformity}(C) = \frac{\max\limits_{l \in L} \left(\sum\limits_{o \in C} [\mathtt{objecttype}(o) = l] \right)}{\mathtt{len}(C)} \quad (4.1)$$

C ist das untersuchte Cluster und ist eine Menge, die die Objekte o des Clusters enthält. L ist die Menge aller Objekttypen, $\mathtt{objecttype}(c)$ gibt den Objekttyp von o im Datensatz zurück. $[P]$ ist die Iverson-Klammer. Das Ergebnis ist 1, wenn P wahr ist, ansonsten 0. ClusterUniformity ist ein Maß für die Einheitlichkeit der Objekttypen in C. Eine Verteilung von 1 gibt an, dass allen Objekten in C der gleiche



(a) Deutscher Datensatz **(b)** Englischer Datensatz

Abbildung 4.2: Ergebnisse der Cluster-Filterung

Objekttyp zugewiesen ist. Ein beliebiges $d \in [1/\mathrm{len}(C), 1]$ gibt an, dass $d \cdot 100\%$ aller Objekte im Cluster der gleiche Objekttyp zugewiesen ist und dass es keinen anderen, häufiger vorkommenden Objekttyp gibt.

Insgesamt wurden im deutschen Datensatz 782 Cluster gefunden. In 592 dieser Cluster wird jeweils genau ein Objekttyp verwendet, d.h. Cluster-Uniformity dieser Cluster ist 1. Im englischen Datensatz weisen 398 von 492 Cluster eine Cluster-Uniformity von 1 auf. Das zeigt, das ähnlichen Objekten in der Regel derselbe Objekttyp zugeordnet wird. Abbildung 4.2 zeigt für die im deutschen und englischen Datensatz gefundenen Cluster jeweils Cluster-Größe und Cluster-Uniformity.

Bei Clustern, welche mindestens zehn Objekte umfassten und für die Cluster-Uniformity mindestens 0,8 beträgt, wurde der Objekttyp aller Objekte an die Klassifizierung der Mehrheit angepasst. In einer Gruppe aus zehn Pin-Spezifikationen, in der ein Objekt als Information klassifiziert wurde, konnte so die Klassifizierung dieses einen Objekts automatisch korrigiert werden. Durch die Wahl eines Schwellenwerts von 0,8 wird erreicht, dass nur bei Clustern mit größtenteils einheitlicher Klassifizierung Änderungen vorgenommen werden. Die Objekte von Clustern mit geringerer Label-Uniformity sind zwar ähnlich, deren Klassifikation aber möglicherweise von Feinheiten abhängig, die nicht vom Clustering-Algorithmus erfasst wurden. Insgesamt wurde die Klassifizierung von 212 Objekten im deutschen Datensatz und 112 Objekten im englischen Datensatz angepasst.

Strukturfilterung

In der Datenmenge sind neben natürlichsprachlichen Sätzen auch Aufzählungen, Formeln und Abbildungen enthalten. Für jede Art von Inhalt können jeweils andere Klassifikationsmodelle die besten Ergebnisse erzielen und nicht alle Modelle können jeden Inhalt klassifizieren. Die verwendeten Klassifikationsmodelle sind in der Lage, beliebigen Text zu klassifizieren. Es wird nicht vorausgesetzt, dass der

SCHRITT	DEUTSCH		ENGLISCH	
	REQ	INF	REQ	INF
Rohdaten	766.769	165.525	634.833	99.479
Nach Vorlagenfilterung	556.696	125.361	406.370	70.840
Nach Reifegradfilterung	95.966	125.361	37.303	70.840
Nach Strukturfilterung	89.373	115.404	33.090	66.473
Nach Duplikatentfernung	37.447	32.120	20.203	17.730
Nach Cluster-Korrektur	37.403	32.164	20.199	17.734

Tabelle 4.1: Datensatzgröße

Eingabetext ein grammatisch korrekter Satz ist. Daher werden alle Textinhalte zur Klassifikation verwendet.

Einige Objekte enthalten Daten in einem eingebetteten Datenformat, welches aufgrund technischer Limitierungen nicht aus der Datenbank extrahiert werden konnte. Dazu gehören unter anderem Abbildungen und bestimmte Tabellen. Alle diese Objekte wurden aus dem Datensatz entfernt und nicht weiter betrachtet.

Vorverarbeitung

Als letztes wurden die folgenden Schritte zur Bereinigung des Datensatzes angewendet.

- Alle Zahlen wurden durch eine Null ersetzt. Für die Klassifikation nach Anforderung und Information ist es nicht relevant, ob beispielsweise eine Komponente 20 mV Spannung benötigt oder 50 mV. Durch diese Änderung wird verhindert, dass der Klassifikator später anhand bestimmter Zahlen Entscheidungen trifft.
- Analog dazu wurden aus demselben Grund Bezeichner von Signalen (beispielsweise `SIG_KLEMME_1`), Komponenten (z.B. `WWC` für Wiper- und Washing-Controller) und anderen Entitäten durch den allgemeinen Bezeichner `IDENTIFIER` ersetzt. Dazu wurden reguläre Ausdrücke verwendet. Diese Ausdrücke finden alle Zeichenketten, die Unterstriche, Ziffern oder Großbuchstaben nicht an erster Stelle enthalten. Mit dieser Methode wird ein Großteil der Bezeichner gefunden.
- Häufig vorkommende Abkürzungen (z.B., insb., u.U., etc.) wurden zur Vereinheitlichung ausgeschrieben.
- Alle Buchstaben wurden in Kleinbuchstaben umgewandelt, damit der Klassifikator nicht zwischen unterschiedlichen Schreibweisen eines Worts (beispielsweise am Satzanfang und in der Mitte eines Satzes) unterscheiden muss.

Nach diesen Schritten enthält der deutsche Datensatz 69.567 Objekte und der englische Datensatz 37.933 Objekte (siehe Tabelle 4.1).

4.4 DATENTRANSFORMATION

Datentransformation mit Word2Vec

Als nächstes wurden die Daten in ein Format überführt, welches mit den Klassifikationsmodellen kompatibel ist. Convolutional-Neural-Networks verwenden wie in Abschnitt 2.8 beschrieben Word-Embeddings als Eingabe. Daher wird ein Word2Vec-Modell benötigt, welches die im Datensatz verwendeten Wörter auf Vektoren abbildet.

Für die englische Sprache existieren vorgefertigte Word2Vec-Modelle, die auf großen Datenmengen trainiert wurden. Das Word2Vec-Modell von Google wurde auf einem Datensatz („Google News“) von 100 Milliarden Wörtern trainiert und umfasst ein Vokabular von drei Millionen Wörtern[2]. Für die Verwendung mit dem erstellten englischen Datensatz ist dieses Modell jedoch nicht geeignet, da in den Anforderungsdokumenten sehr viel Fachvokabular verwendet wird, welches in dem Datensatz von Google nicht vorkommt. Daher wurden zwei neue Word2Vec-Modelle speziell für die in dieser Arbeit verwendeten deutschen und englischen Anforderungsdokumente trainiert.

Als Erstes wurde für das Training der beiden Word-Embeddings jeweils ein Korpus erstellt. Für den jeweiligen Korpus wurden sämtliche in dessen Sprache zur Verfügung stehenden Anforderungen und Informationen verwendet. Keine der in Abschnitt 4.2 beschriebenen Filtertechniken wurde angewendet, da für das Training der Word2Vec-Modelle die Qualität der Daten in Hinblick auf die Eignung zur Klassifikation in Anforderungen und Informationen nicht relevant ist. Lediglich die Vorverarbeitungsschritte wurden angewendet, damit das Wörterbuch des Word2Vec-Modells zum jeweiligen Datensatz passt. Insgesamt stehen für den deutschen Korpus 20.407.543 Wörter zur Verfügung, für den englischen Korpus sind es 32.978.683 Wörter.

Für das Training der Modelle wurden anfangs folgende Hyperparameter festgelegt: 128 Dimensionen für die Wort-Vektoren, Fensterbreite 5, minimale Dokumentfrequenz 0,001 und 20 Trainingsiterationen. Modelle mit diesen Hyperparametern stellen eine gute Ausgangsbasis dar, da sie einen Großteil der in den Datensätzen vorkommenden Wörter abdecken und die Leistung des Klassifikator nicht durch zu kleine Wort-Vektoren limitiert wird. In der Evaluation (siehe Abschnitt 6.1) wird gezielt nach den besten Hyperparametern für den erstellten Datensatz gesucht.

Die erzeugten Wort-Vektoren weisen typische Eigenschaften von Word2Vec-Modellen auf. Die Vektoren von Wörtern, die in ähnlichen Kontexten verwendet werden, haben eine geringere Distanz zueinander als Wörter, die nichts miteinander zu tun haben. Abbildung 4.3 zeigt einen mit t-SNE [73] visualisierten Ausschnitt des deutschen Word2Vec-Modells. In dieser Grafik sind mehrere Gruppen von Wörtern erkennbar: Wörter, die mit Signalen verwandt sind (lin, can,

2 https://code.google.com/archive/p/word2vec/

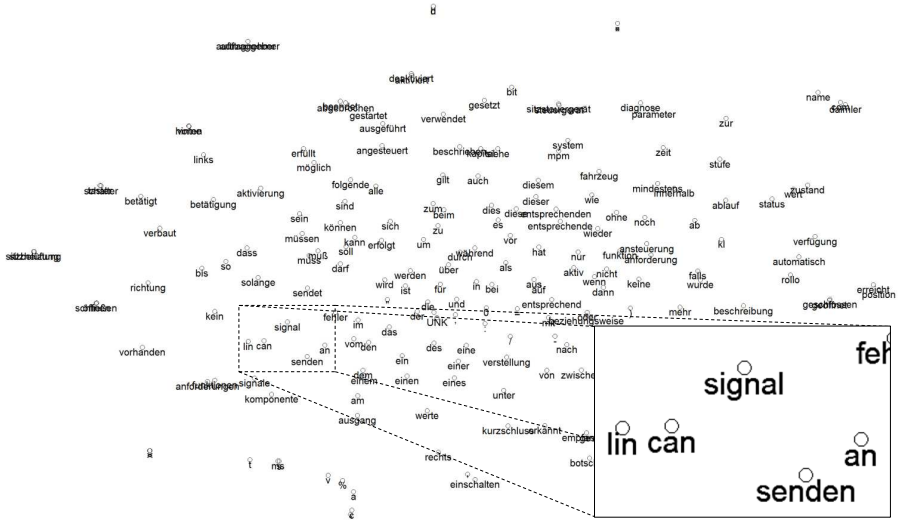


Abbildung 4.3: t-SNE-Visualisierung des deutschen Word2Vec-Modells

senden, signal, signale), Modalverben (kann, soll, muss, muß, müssen), Einheiten (t, ms, v, %, a) und Paare entgegengesetzter Wörter (öffnen/schließen, aktiviert/deaktiviert, vorne/hinten). Diese Wörter haben jeweils ähnliche Wortvektoren und erlauben es damit dem Klassifikationsmodell, besser über diese Wörter zu generalisieren.

Bag-of-Words

Zum Training der Vergleichsklassifikatoren wird anstatt eines Word2Vec-Modells ein Modell benötigt, welches jedes Objekt auf einen Vektor mit einer stets gleichen Länge abbildet. Dafür bietet sich das Bag-of-Words-Modell [42, S. 65] an.

Das Bag-of-Words-Modell wurde auf Basis des gleichen Korpus erstellt, der auch für das Training des Word2Vec-Modells verwendet wurde. Die minimale Dokumentfrequenz wurde auf 0,001 gesetzt. Das resultierende deutsche Wörterbuch enthält 1.982 Wörter (Englisch: 2.073) und erlaubt es, beliebige Lastenheftobjekte in Vektoren umzuwandeln. Weiterhin wurde Principal-Component-Analysis eingesetzt, um die Dimensionalität des Feature-Raums auf 100 zu reduzieren.

Bag-of-N-Grams, Zeichen-basiert

Ein weiteres Klassifikationsmodell wurde nicht auf Wortebene, sondern auf Zeichenebene trainiert. Für dieses Modell werden als Features keine Wörter verwendet, sondern Folgen (auch *N-Grams*) von Zeichen bis zu einer maximalen Länge von fünf Zeichen. Auch hier wurde die minimale Dokumentfrequenz auf 0,001 gesetzt, wodurch der Feature-Raum des englischen Datensatzes 12.260 und der Feature-Raum des deutschen Datensatzes 15.450 unterschiedliche Zeichenketten umfasst. Mittels PCA wurden beide Feature-Räume auf 1.000 Features reduziert.

4.5 AUSWAHL EINES KLASSIFIKATORS

Grundsätzlich wird bei der Klassifikation von Text zwischen verschiedenen Arten von Klassifikatoren unterschieden. Zeichen-basierte Klassifikatoren benutzen einzelne Zeichen und Zeichenfolgen des Eingabetextes zur Klassifikation. Analog dazu benutzen Wort-basierte Klassifikationen Wörter und Wortfolgen zur Klassifikation.

Wie in Abschnitt 3.5 beschrieben entscheidet öfters die Verwendung bestimmter Wörter über die Klassifizierung eines Objekts. Daher werden besonders Wort-basierte Klassifikatoren betrachtet. Die folgenden Klassifikationsmodelle werden in dieser Arbeit miteinander verglichen. Grundsätzlich eignen sich alle diese Modelle zur Klassifikation von Text; durch ihre unterschiedliche Funktionsweise erzielen sie jedoch unterschiedlich gute Ergebnisse auf den Datensätzen.

- Trivialer Wort-basierter Klassifikator
- Naïve-Bayes
- Entscheidungsbäume
- Support-Vector-Machines
- Logistische Regression
- Logistische Regression, Zeichen-basiert
- Feed-Forward-Netzwerk
- Convolutional-Neural-Network

Alle diese Klassifikationsmodelle wurden auf beide Datensätze angewendet. Auf Basis dieser Ergebnisse wird derjenige Klassifikator mit den besten Ergebnissen für den weiteren Verlauf der Arbeit ausgewählt.

4.6 ANWENDUNG DER KLASSIFIKATOREN

Datensatz-Balance

In beiden Datensätzen existieren mehr Anforderungen als Informationen. Bei den meisten Klassifikationsmodellen führt dies zu Problemen, da ein Modell bei einem nicht ausbalancierten Datensatz in der Regel dazu neigt, primär die dominante Klasse vorherzusagen.

Für die Modelle Naïve-Bayes, Entscheidungsbäume, Support-Vector-Machines und logistische Regression wurde Oversampling eingesetzt: zufällig ausgewählte Beispiele der unterrepräsentierten Klasse *Information* wurden während des Trainings dupliziert, bis ein Gleichgewicht hergestellt wurde.

	DEUTSCH	ENGLISCH
Anforderung	1,860	1,878
Information	2,163	2,139

Tabelle 4.2: Klassengewichte zum Ausgleich des Klassenungleichgewichts

Die Modelle Feed-Forward-Netzwerk und Convolutional-Neural-Network werden mit einem auf Stochastic-Gradient-Descent basierenden Optimierungsalgorithmus trainiert, bei dem Klassengewichte angegeben werden können [110]. Für jede Klasse beeinflusst ein zugehöriges Klassengewicht, wie schnell das Modell von Beispielen dieser Klasse lernt. Indem unterrepräsentierten Klassen höhere Klassengewichte zugeordnete werden, wird die Klassenungleichheit ausgeglichen. Die Klassengewichte w_c für eine Klasse c in einem Datensatz DS werden wie folgt berechnet:

$$w_c = \frac{\mathrm{len}(DS)}{\sum_{o \in DS} [\mathrm{ot}(o) = c]} \tag{4.2}$$

$\mathrm{ot}(o)$ gibt den Objekttyp des Objekts o zurück. Die für beide Datensätze berechneten Klassengewichte befinden sich in Tabelle 4.2.

Klassifikatorkonstruktion

Die Klassifikatoren wurden wie folgt konstruiert. Für die meisten Klassifikatoren wurden Standardimplementierungen aus der Python-Bibliothek Scikit-Learn[3] verwendet.

TRIVIALER WORT-BASIERTER KLASSIFIKATOR: Dieses Modell klassifiziert Objekte einer Spezifikation als Anforderungen, wenn deren Text bestimmte Schlüsselwörter enthält. Für den deutschen Datensatz sind dies die Wörter „muss“, „muß“, „müssen“, „soll“, „sollen“. Für den englischen Datensatz wurden „must“ und „shall“ ausgewählt. Objekte, die diese Wörter nicht enthalten, werden als Information klassifiziert. Dieser Klassifikator soll zeigen, wie effektiv ein trivialer Ansatz Anforderungen von Informationen unterscheiden kann und ob andere Modelle durch das Erlernen von Mustern besser klassifizieren können.

NAÏVE-BAYES: Ein multinomialer Naïve-Bayes-Klassifikator, der für ein Objekt jeder Klasse Wahrscheinlichkeiten zuordnet. Keine besonderen Einstellungen.

ENTSCHEIDUNGSBAUM: Ein Entscheidungsbaum-Klassifikationsmodell. Als Kriterium zur Auswahl der für die Knoten verwendeten Prädikate wird der Gini-Index verwendet.

3 https://scikit-learn.org

SUPPORT-VECTOR-MACHINE: Ein lineares Support-Vector-Machine-Modell. Ein nicht-linearer Kernel wurde nicht eingesetzt, da deren Verwendung zu keinen besseren Ergebnissen führte und lediglich die für das Training benötigte Zeit verlängerte.

LOGISTISCHE REGRESSION: Für diesen Klassifikator wurde die Standard-Implementierung von Scikit-Learn ohne Anpassung der Parameter verwendet.

FEED-FORWARD-NETZWERK: Dieses Netzwerk wurde mit der Python-Bibliothek Keras[4] implementiert. Das Netzwerk besteht aus einer Eingabeebene, dessen Länge der Anzahl an Wörter im Wörterbuch des jeweiligen Datensatzes entspricht, einer versteckten Ebene mit 50 Neuronen, und einer Ausgabeebene mit zwei Neuronen, die nach Anwendung der Softmax-Funktion Wahrscheinlichkeiten für beide Klassen anzeigen. Als Aktivierungsfunktion wurden Rectified-Linear-Units verwendet. Nach mehreren Versuchen mit unterschiedlichen Hyperparametern (Anzahl und Größe der versteckten Ebenen, Aktivierungsfunktion) stellte sich heraus, dass bereits eine versteckte Ebene zu großen Verbesserungen in der Genauigkeit der Klassifikation führt. Die Verwendung von zwei oder mehr versteckten Ebenen führte zu keinen weiteren Verbesserungen, sondern verlangsamte lediglich das Training des Modells.

LOGISTISCHE REGRESSION – ZEICHEN-BASIERT: Logistische Regression lieferte auf beiden Datensätzen die besten Ergebnisse. Ein weiteres Modell wurde auf Zeichenebene trainiert, um zu untersuchen, ob dies zu anderen oder besseren Ergebnissen führt. Bei der Feature-Generierung wurden Zeichenketten der Länge 2 bis 5 in das Wörterbuch aufgenommen.

CONVOLUTIONAL-NEURAL-NETWORK: Dieses Netzwerk wurde ebenfalls mit Keras implementiert. Die Eingabedaten werden mit den vortrainierten Word2Vec-Modellen in Dokumentmatrizen transformiert. Als maximale Satzlänge wurde 50 festgelegt. Wie in Abbildung 4.1 ersichtlich wird dadurch ein Großteil der Objekte in den Datensätzen vollständig abgedeckt. Das Netzwerk besteht wie in Abschnitt 2.8 beschrieben aus einer Convolution-Ebene, einer Pooling-Ebene und einer vollständig verknüpften Ebene. Für die Convolution-Ebene wurden jeweils 32 Filter der Länge 2, 3 und 4 verwendet. Als Aktivierungsfunktion wurde die ReLU-Funktion verwendet. Wie auch bei dem Feed-Forward-Netzwerk sind dies die Einstellungen, die nach mehreren Versuchen mit unterschiedlichen Einstellungen zu den besten Ergebnissen geführt haben.

4 https://keras.io

	DEUTSCH		ENGLISCH	
	REQ	INF	REQ	INF
TRIVIALER WORT-BASIERTER KLASSIFIKATOR				
Precision	0,653	0,601	0,675	0,691
Recall	0,795	0,422	0,864	0,421
Genauigkeit		0,637		0,679
NAÏVE-BAYES				
Precision	0,803	0,732	0,817	0,801
Recall	0,805	0,729	0,869	0,730
Genauigkeit		0,773		0,811

Tabelle 4.3: Ergebnisse der getesteten Klassifikationsmodelle

Evaluation der Klassifikatoren mit Kreuzvalidierung

Daraufhin wurde jeder Klassifikator mit dem Kreuzvalidierungsverfahren trainiert und getestet. Bei diesem Verfahren wird die Datenmenge in eine beliebige Anzahl n gleich großer Teile unterteilt (hier: $n = 10$). Daraufhin wird in insgesamt n Durchläufen jeweils ein anderer Teil beiseitegelegt und die verbleibenden $n - 1$ Teile zum Trainieren verwendet. Oversampling wurde jeweils nur auf den Trainingsdatensatz angewendet. Die Leistung des Modells wird an dem beiseitegelegten Teil gemessen. In jedem Durchlauf wird ein neues Modell trainiert, welches nach dem Testen verworfen wird. Nach allen Durchläufen wurden die Ergebnisse aller n Durchläufe gemeinsam ausgewertet. In Tabelle 4.3 sind die Ergebnisse aller Klassifikatoren auf beiden Datensätzen dargestellt.

Bei allen Klassifikatoren sind die Ergebnisse auf dem englischen Datensatz besser als die Ergebnisse auf dem deutschen Datensatz. Dieses Phänomen kann zwei Ursachen haben. Zum einen können englischsprachige Anforderungen und Informationen aufgrund bestimmter Merkmale der englischen Sprache besser voneinander unterscheidbar sein. Andererseits ist es möglich, dass die Qualität des englischen Datensatzes bezüglich der Präzision der verwendeten Formulierungen und der Genauigkeit der Klassifikation besser als die Qualität des deutschen Datensatzes ist.

Der triviale Wort-basierte Klassifikator zeigt, dass die Identifikation von Anforderungen anhand bestimmter Schlüsselwörter bereits zu einem Recall auf Anforderungen führt, der zwar geringer als der Recall der anderen Modelle ist, aber dennoch mit diesen vergleichbar ist. Die weitaus geringere Precision von Anforderungen auf beiden Datensätzen zeigt allerdings, dass durch dieses Vorgehen zu viele Objekte als Anforderungen klassifiziert werden. Precision und Recall auf Informationen fällt für diesen Ansatz auf beiden Datensätzen sehr schlecht aus.

	DEUTSCH		ENGLISCH	
	REQ	INF	REQ	INF
ENTSCHEIDUNGSBAUM				
Precision	0,830	0,765	0,860	0,812
Recall	0,828	0,769	0,865	0,805
Genauigkeit		0,803		0,840
SUPPORT-VECTOR-MACHINE				
Precision	0,815	0,788	0,868	0,826
Recall	0,856	0,734	0,876	0,815
Genauigkeit		0,804		0,850
LOGISTISCHE REGRESSION				
Precision	0,819	0,788	0,869	0,833
Recall	0,854	0,741	0,882	0,816
Genauigkeit		0,806		0,855
FEED-FORWARD-NETZWERK				
Precision	0,815	0,724	0,865	0,812
Recall	0,790	0,755	0,865	0,813
Genauigkeit		0,775		0,843
LOGISTISCHE REGRESSION, ZEICHEN-BASIERT				
Precision	0,828	0,790	0,866	0,820
Recall	0,853	0,757	0,871	0,813
Genauigkeit		0,812		0,847
CONVOLUTIONAL-NEURAL-NETWORK				
Precision	0,865	0,832	0,901	0,867
Recall	0,880	0,812	0,905	0,862
Genauigkeit		0,851		0,887

Tabelle 4.3 (fortges.): Ergebnisse der getesteten Klassifikationsmodelle

Alle Klassifikatoren, bei denen zur Transformation der Daten das Bag-of-Words-Modell zum Einsatz kamen, zeigen ähnliche Ergebnisse. Hinsichtlich der Genauigkeit schnitt das Naïve-Bayes-Modell am schlechtesten und logistische Regression am besten ab. Das Zeichenbasierte Modell der logistischen Regression zeigt, dass die Verwendung von Zeichenketten statt Wörtern zu keinen besseren Ergebnissen in der Klassifikation führt.

Das CNN-Klassifikationsmodell lieferte auf beiden Datensätzen die besten Ergebnisse.

5 ERWEITERUNG DES ANSATZES

In diesem Abschnitt wird gezeigt, wie die Leistung des Klassifikationsmodells mit unterschiedlichen Techniken weiter verbessert wird. In Abschnitt 5.1 wird die Genauigkeit der Klassifikation durch die zusätzliche Verwendung von Kontextinformationen weiter gesteigert. Abschnitt 5.2 stellt einen Ansatz vor, der die Entscheidungen von CNNs bei der Klassifikation von Text visuell erklärt.

5.1 KLASSIFIKATION MIT KONTEXT

Der auf Convolutional-Neural-Networks basierende Klassifikator benutzt zur Klassifikation der Objekte lediglich die Wörter des jeweiligen Objekts. Wie in Abschnitt 3.4 und 3.5 ausgeführt besitzen Objekte in Lastenheften deutlich mehr Informationen als lediglich den Text des jeweiligen Objekts. Jedes Objekt hat weitere Attribute, die jeweils bestimmte Eigenschaften des Objekts festlegen. Jedes Objekt ist weiterhin von anderen Objekten umgeben und deren Text und Attribute können ebenfalls zur Klassifikation verwendet werden. Dadurch ist es möglich, Objekte richtig zu klassifizieren, die ohne diese zusätzlichen Kontextinformationen nicht korrekt klassifiziert werden können.

Beispiele schwierig zu klassifizierender Objekte

In den Datensätzen existieren Objekte, die von keinem der in Kapitel 4 getesteten Klassifikatoren richtig klassifiziert werden konnten. Einige dieser Objekte sind in Tabelle 5.1 zusammen mit dem Text von Elternobjekt, Vorgängerobjekt und Nachfolgerobjekt aufgelistet. Das zu klassifizierende Objekt ist jeweils kursiv geschrieben. Die jeweils korrekte Klassifikation des Objekts steht in eckigen Klammern.

In Beispiel 1 und 2 wird jeweils ein Signal beschrieben. In beiden Fällen handelt es sich nicht um eine Anforderung, da lediglich die Bedeutung der Signale innerhalb der Komponente skizziert wird. Insbesondere im zweiten Beispiel ist dies durch die Überschrift offensichtlich, obwohl dieses Objekte Formulierungen enthält, die es eher als Anforderung kennzeichnen („Die Komponente muss...“).

Das dritte Beispiel spezifiziert eine Eigenschaft eines Systems (Farbe) und ist deshalb eine Anforderung. In Beispiel 4 wird die Schnittstelle einer Komponente durch die Auflistung relevanter Signale spezifiziert. Das fünfte Beispiel zeigt eine Auflistung von Zuständen, welche die Verfügbarkeit einer bestimmten Funktion steuern. In allen Fällen ist nur unter zusätzlicher Betrachtung des Kontextes der Objekte (Überschrift) ersichtlich, dass es sich bei diesen Objekten um Anforderungen handelt. Diese Art der Spezifikation wird in den Datensätzen häufig

1	SIGNAL: XYZ_01 *Modus des Fahrzeugkoordinators.* [Information] Eine detaillierte Beschreibung der Signalerstellung befindet sich in der DOORS-Anforderungsspezifikation.
2	SIGNALBESCHREIBUNG *Mit diesem Signal wird die Aktivierung des Fehlermanagements im Fahrzeugsteuergerät angezeigt. Die Komponente muss dieses Signal benutzen, um die Diagnoseanforderungen zu erfüllen.* [Information]
3	Die Oberfläche des Paneels muss in einer bestimmten Technologie implementiert werden: *Farbe: „Gray-Surface" gemäß Dokument x* [Anforderung] Technologie: Special-Form
4	Die folgenden Signale müssen von Komponente ABC empfangen werden: offset of vehicle yaw rate unfiltered / unadjusted *offset of vehicle longitudinal acceleration* [Anforderung] offset of vehicle lateral acceleration
5	Die Schaltfläche in der Head-Unit muss ausgegraut werden, wenn die Funktion aufgrund folgender anderen aktiven Funktionen oder Zustände nicht aktiviert werden kann: Fahrzeuggeschwindigkeit > 0 *Funktion defekt oder Steuergerätefehler* [Anforderung] Steuergerät im Diagnosemodus
6	ÜBERBLICK UND KURZE BESCHREIBUNG *Die Komponente muss durch unterschiedliche Bedingungen aktivierbar sein.* [Information] Die Aktivierungs- und Deaktivierungsbedingungen werden in anderen Softwareblöcken kontrolliert und werden im Controller der Schalteraktivierung kombiniert.

Tabelle 5.1: Beispiele falsch klassifizierter Objekte. Das zu klassifizierende Objekt ist jeweils kursiv geschrieben. Die korrekte Klassifizierung dieser Objekte befindet sich in den Klammern.

und in vielen anderen Situation verwendet (siehe Abschnitt 3.5), da dadurch ohne Verlust von Lesbarkeit und Eindeutigkeit viel Text eingespart werden kann. Viele der falsch klassifizierten Objekte in den Datensätzen sind ähnlich zu diesen drei Beispielen.

Beispiel 6 wurde durch die Formulierung „Die Komponente muss..." als Anforderung klassifiziert. Dieses Objekt ist allerdings Teil eines Kapitels, welches informell die grobe Funktionalität der Komponente beschreibt und sollte daher als Information klassifiziert werden. Wenn der Kontext eines Objekts in die Klassifikation mit einbezogen wird, können Objekte wie dieses möglicherweise korrekt klassifiziert werden.

Aus diesen Beispielen lässt sich ableiten, dass besonders die Überschrift, bzw. das Kapitel, unter dem ein Objekt steht für die Klassifikation relevant ist. Diese Vermutung wird im Folgenden genauer untersucht.

5.1.1 Konstruktion eines Klassifikators mit Kontextunterstützung

Basierend auf den Ergebnissen aus Kapitel 4 wird nun die ein Klassifikationsmodell entwickelt, welches neben dem Text eines Objekts auch Kontextinformationen in die Klassifikation mit einbezieht. Dazu werden wiederum die neun Schritte des KDD-Prozesses durchlaufen und an notwendigen Stellen Änderungen vorgenommen.

Erweiterung des Datensatzes um Kontextinformationen

Die Datensätze müssen die erforderlichen Kontextinformationen enthalten und daher angepasst werden. Grundsätzlich bleibt der in Abschnitt 4.2 und 4.3 beschriebene Ablauf unverändert, damit die Ergebnisse des Klassifikators mit Kontext mit den Ergebnissen der Klassifikatoren ohne Kontext vergleichbar sind. Jedes Objekt in beiden Datensätzen wird lediglich um die folgenden Informationen erweitert:

- Text des Elternobjekts, falls vorhanden.
- Text des Vorgängerobjekts, falls vorhanden.
- Text des Nachfolgerobjekts, falls vorhanden.

Auf den Text dieser Objekte werden jeweils die gleichen Vorverarbeitungsschritte angewendet, die auch auf den Text des eigentlichen Objekts angewendet wurden: Zahlen wurden durch Nullen ersetzt, Bezeichner wurden durch `IDENTIFIER` ersetzt, Abkürzungen wurden ausgeschrieben und sämtliche Buchstaben wurden durch Kleinbuchstaben ersetzt.

Jedes Objekt der Datensätze besteht nun aus insgesamt vier *Komponenten* zuzüglich des Objekttyps. Weitere Attribute der Objekte wie beispielsweise Reifegrad oder Empfohlene Verifikationsmethode wurden nicht berücksichtigt, da diese in keinem erkennbaren Zusammenhang mit der Objekttyp-Klassifikation eines Objekts stehen.

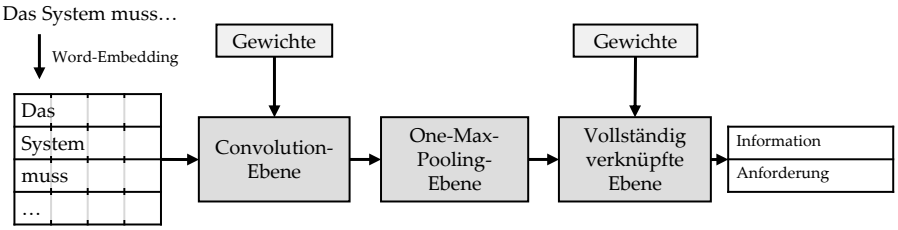


Abbildung 5.1: CNN-Klassifikator ohne Kontext

Erweiterung des Klassifikationsmodells

Das Convolutional-Neural-Network des Klassifikators ist in der aktuellen Form in Abbildung 5.1 vereinfacht dargestellt. Das Netzwerk kann in der aktuellen Form nicht zur Klassifikation von Objekten mit Kontext verwendet werden, da es nur einen Eingang für eine Wortfolge besitzt. Es wäre möglich, die einzelnen Komponenten eines zu klassifizierenden Objekts zu konkatenieren und so den Klassifikator weiterzuverwenden. In diesem Fall geht jedoch die Information verloren, aus welcher Komponente eine bestimmte Wortfolge stammt. Da dies für die Klassifikation mit Kontext relevant ist, muss das Netzwerk angepasst werden.

Abbildung 5.2 zeigt das modifizierte Netzwerk. Anstatt nur eine Wortfolge als Eingabe zu akzeptieren, akzeptiert das Netzwerk nun jeweils eine Wortfolge für jede Komponente. Die in Abbildung 5.1 gezeigten Schritte von Word-Embedding bis One-Max-Pooling werden auf jede Komponente separat angewendet. Die resultierenden Feature-Vektoren aller Komponenten werden konkateniert. Durch eine vollständig verknüpfte Ebene werden daraufhin bestimmte Features mit den Ausgabeklassen assoziiert. An dieser Stelle ist das Netzwerk noch immer in der Lage, Aktivierungen im Feature-Vektor auf die zugehörige Komponente zurückzuführen. So ist das Netzwerk in der Lage, Objekte durch Kombination von Features aus verschiedenen Komponenten zu klassifizieren.

Die Gewichte der komponentenspezifischen Operationen werden zwischen allen Komponenten geteilt. Dies bedeutet, dass bei jeder Komponente dasselbe Word-Embedding-Modell für die Transformation der Eingabedaten verwendet wird und das alle Convolution-Ebenen erlernen, dieselben Muster zu erkennen. Die vollständig verknüpfte Ebene besitzt für die Feature-Vektoren jeder Komponente separate Gewichte und kann deshalb je nachdem in welcher Komponente ein bestimmtes Muster erkannt wurde ein Objekt unterschiedlich klassifizieren. Durch die geteilten Gewichte wird die Netzwerkgröße reduziert und die Klassifikationsergebnisse sind besser mit den Ergebnissen der Klassifikation ohne Kontext vergleichbar, da die Anzahl der Gewichte nicht mit der Anzahl der Komponenten steigt[1].

Training des Klassifikators mit Kontextinformationen

Der Klassifikator wurde daraufhin auf den Datensätzen mit Kontextinformationen trainiert. Es wurden die gleichen Hyperparameter

1 In der Regel führen mehr Gewichte zu leicht besseren Ergebnissen.

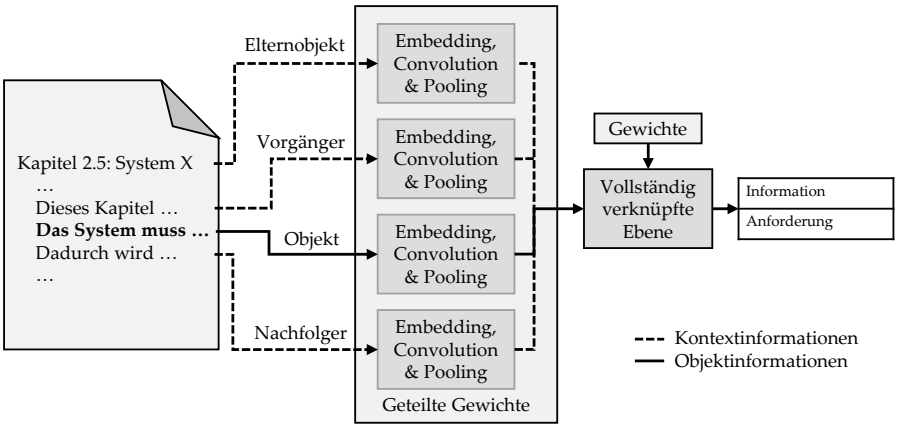


Abbildung 5.2: CNN-Klassifikator mit Kontext

HYPERPARAMETER	WERT
Größe der Wortvektoren	128
Filterlängen	2, 3, 4
Filteranzahl je Länge	32
Satzlänge je Komponente	50

Tabelle 5.2: Hyperparameter für die Klassifikation mit Kontext

ausgewählt, die auch bereits für den CNN-Klassifikator ohne Kontextinformationen eingesetzt wurden. Diese sind noch einmal in Tabelle 5.2 zusammengefasst. Für das Training wurde erneut zehnfache Kreuzvalidierung verwendet.

5.1.2 Ergebnisse der Klassifikation mit Kontext

Die Ergebnisse der Klassifikation mit Kontext sind in Tabelle 5.3 dargestellt. Insgesamt zeigen die verwendeten Metriken Precision, Recall und Genauigkeit deutliche Verbesserungen bei der Verwendung von Kontextinformationen. Mithilfe der Formel für relative Veränderung

$$r = \frac{\text{Absolute Änderung}}{\text{Referenzpunkt}} \tag{5.1}$$

$$= \frac{\text{Acc}_{after} - \text{Acc}_{before}}{1 - \text{Acc}_{before}} \tag{5.2}$$

wird auf Basis der Genauigkeit ermittelt, um wie viel Prozent die Menge falsch klassifizierter Objekte kleiner wird. Diese Metrik berücksichtigt ebenfalls, dass eine Verbesserung der Genauigkeit um 1% von 98% auf 99% mit einem größeren Aufwand verbunden ist als eine Verbesserung der Genauigkeit um 1% von 88% auf 89%. Auf dem deutschen Datensatz wird die Menge falsch klassifizierter Objekte

DEUTSCH	ANFORDERUNG	INFORMATION
Precision	$0,865 \rightarrow 0,905$	$0,832 \rightarrow 0,883$
Recall	$0,880 \rightarrow 0,915$	$0,812 \rightarrow 0,869$
Genauigkeit	$0,851 \rightarrow 0,896$	
ENGLISCH	ANFORDERUNG	INFORMATION
Precision	$0,901 \rightarrow 0,923$	$0,867 \rightarrow 0,910$
Recall	$0,905 \rightarrow 0,937$	$0,862 \rightarrow 0,891$
Genauigkeit	$0,887 \rightarrow 0,918$	

Tabelle 5.3: Verbesserungen durch die Verwendung von Kontext

KONTEXT	DEUTSCH	ENGLISCH
Kein Kontext	0,851	0,887
Vorgängerobjekt	0,866	0,890
Nachfolgerobjekt	0,865	0,890
Elternobjekt	0,886	0,914
Vollständig	0,896	0,918

Tabelle 5.4: Genauigkeit mit unterschiedlichen Kontextkomponenten

um 30,2% reduziert, auf dem englischen Datensatz sind es 27,4%. Die Verwendung von Kontext führt dementsprechend sowohl auf dem englischen als auch auf dem deutschen Datensatz zu vergleichbaren Verbesserungen.

Einfluss einzelner Komponenten auf die Genauigkeit

Weiterhin wurde untersucht, welchen Einfluss die einzelnen Komponenten in Isolation auf die Genauigkeit der Klassifikation haben. Dazu wurde der Klassifikator mit jeweils nur dem Objekttext und einer der drei Komponenten Elternobjekt, Vorgängerobjekt und Nachfolgerobjekt trainiert. Die Ergebnisse befinden sich in Tabelle 5.4. Allein durch die Verwendung des Elternobjekts im Kontext für die Klassifikation wird bereits fast die Genauigkeit erreicht, die bei der Verwendung des vollständigen Kontextes erzielt wird. Die Klassifikation mit Vorgängerobjekt und Nachfolgerobjekt allein hat keine großen Auswirkungen auf die Genauigkeit der Klassifikation und die Genauigkeit steigt durch die Verwendung von Vorgängerobjekt und Nachfolgerobjekt zusätzlich zum Elternobjekt nur gering. Diese Beobachtungen können sowohl auf dem deutschen als auch auf dem englischen Datensatz gemacht werden.

Insgesamt bestätigen die Ergebnisse die Vermutung, dass besonders die Überschriften eines Objekts für die Klassifikation relevant sind. Allerdings existieren auch wenige Objekte in den Datensätzen, die

erst durch die zusätzliche Verwendung von Vorgängerobjekt und Nachfolgerobjekt richtig klassifiziert werden konnten.

5.1.3 Zusammenfassung

Insgesamt werden durch die Einbeziehung von Kontextinformationen in die Klassifikation von Anforderungen und Informationen wesentliche Verbesserungen in der Klassifikationsgenauigkeit erzielt. Diese Verbesserungen beeinflussen die Anwendbarkeit der Klassifikation in einem Werkzeug positiv, da solche Werkzeuge durch die Verbesserungen weniger False-Positives produzieren werden.

5.2 BEREITSTELLEN VON ERKLÄRUNGEN

Neuronale Netzwerke sind sehr gut darin, komplexe Muster in großen Datenmengen zu erkennen und diese Muster auf unbekannte Beispiele zu übertragen. Dies wurde in den vorangegangenen Abschnitten anhand der Klassifikation von Lastenheftinhalten in Anforderungen und Informationen bereits gezeigt. Eine Schwäche neuronaler Netzwerke ist jedoch, dass sie sich wie eine Black-Box verhalten. Wenn ein neuronales Netzwerk ein Beispiel richtig klassifiziert, ist nicht ersichtlich, warum dieses Beispiel seiner Klasse angehört. Weiterhin ist es bei falsch klassifizierten Beispielen schwierig nachzuvollziehen, warum das Netzwerk an diesem Beispiel scheitert. Dies erschwert sowohl das Finden und Ausbessern von Fehlern durch den Netzwerkentwickler als auch die Einsetzbarkeit eines neuronalen Netzwerkes in einem Werkzeug, da die Nutzer eines solchen Werkzeugs diesem nicht vertrauen, wenn sie dessen Entscheidungen nicht verstehen [41]. Daher ist es notwendig, dass Softwarewerkzeuge einfache und schnell verständliche Erklärungen für getroffene Entscheidungen bereitstellen. In diesem Abschnitt wird vorgestellt, wie die Entscheidungen von Convolutional-Neural-Networks bei der Klassifikation von Text erklärt werden können.

Neuronale Netzwerke sind Black-Boxen

Erklärungen von Entscheidungen neuronaler Netzwerke verfolgen zwei Ziele [41]:

Interpretierbare und vollständige Erklärungen für Textklassifikation

INTERPRETIERBARKEIT: Die internen Faktoren, die zu einer Entscheidung eines neuronalen Netzwerks geführt haben, müssen durch die Erklärung so präsentiert werden, dass sie von einem Menschen verstanden werden können.

VOLLSTÄNDIGKEIT: Die Erklärung muss die Funktion des neuronalen Netzwerks möglichst genau beschreiben. Eine Erklärung ist vollständiger, wenn sie die Vorhersage des Netzwerkverhaltens in mehr Situationen erlaubt. Die vollständigste Erklärung eines

> neuronalen Netzwerks ist die Offenlegung aller mathematischen Operationen, die zu einer Entscheidung geführt haben.

Die Schwierigkeit bei der Erstellung von Erklärungen liegt in der Balance zwischen Interpretierbarkeit und Vollständigkeit. Analog zur Balance zwischen Recall und Precision ist es schwierig, sowohl hohe Interpretierbarkeit als auch hohe Vollständigkeit zu erreichen. Eine vollständige Erklärung ist nur selten einfach verständlich, eine verständliche Erklärung lässt häufig wichtige Details aus.

Das in dieser Arbeit eingesetzte Convolutional-Neural-Network klassifiziert Objekte anhand von bestimmten Wörtern und Wortgruppen, die häufig in den Beispielen der jeweiligen Klassen auftreten. Kommen in einem zu klassifizierenden Objekt die Wörter und Wortgruppen einer Klasse exakt oder in verwandter Form vor, wird es meist dieser Klasse zugeordnet. Eine interpretierbare Erklärung für CNNs zeigt an, welche Wörter und Wortgruppen zur Entscheidung geführt haben.

Unterschiedliche Teile eines Objekts können jedoch unterschiedlich großen Einfluss auf das Ergebnis der Klassifikation haben. Weiterhin kann es für die Klassifikation entscheidend sein, dass ein Objekt eine Kombination bestimmter Wortgruppen enthält. Eine vollständige Erklärung zeigt alle diese Zusammenhänge auf und wird dadurch sehr umfangreich.

Das CNN wird in einem Werkzeug eingesetzt, welches von den Autoren von Anforderungsdokumenten verwendet wird. Daher ist es wichtig, dass dieses vorrangig einfach interpretierbar ist. Im folgenden Abschnitt wird daher ein Verfahren vorgestellt, welches visuelle Erklärungen für die individuelle Klassifikationsergebnisse von CNNs erstellt. Das folgende Beispiel zeigt eine Erklärung für ein Objekt, welches durch das Netzwerk als Anforderung klassifiziert wurde:

> Die Zeitdauer bis der Schalter als hängend erkannt wird , muss über die Diagnose einstellbar sein .

In diesen Erklärungen werden diejenigen Wörter hervorgehoben, die einen besonders großen Einfluss auf die Klassifikationsentscheidung haben. Je stärker die Hervorhebung, umso größer der Einfluss auf das Ergebnis. In der obigen Anforderung haben die Wörter „als hängend erkannt“ und „muss über“ den größten Einfluss auf die Entscheidung des CNNs, dieses Beispiel als Anforderung zu klassifizieren.

Diese Erklärung ist interpretierbar, da auf einen Blick erkannt werden kann, welchen Einfluss welcher Teil des Satzes auf die Klassifikation hat. Für diese kompakte Darstellungsform müssen allerdings Abstriche bezüglich der Vollständigkeit gemacht werden: Es ist beispielsweise nicht erkennbar, ob zwei benachbarte hervorgehobene Wörter unabhängig voneinander oder nur in Kombination miteinander das Ergebnis beeinflussen.

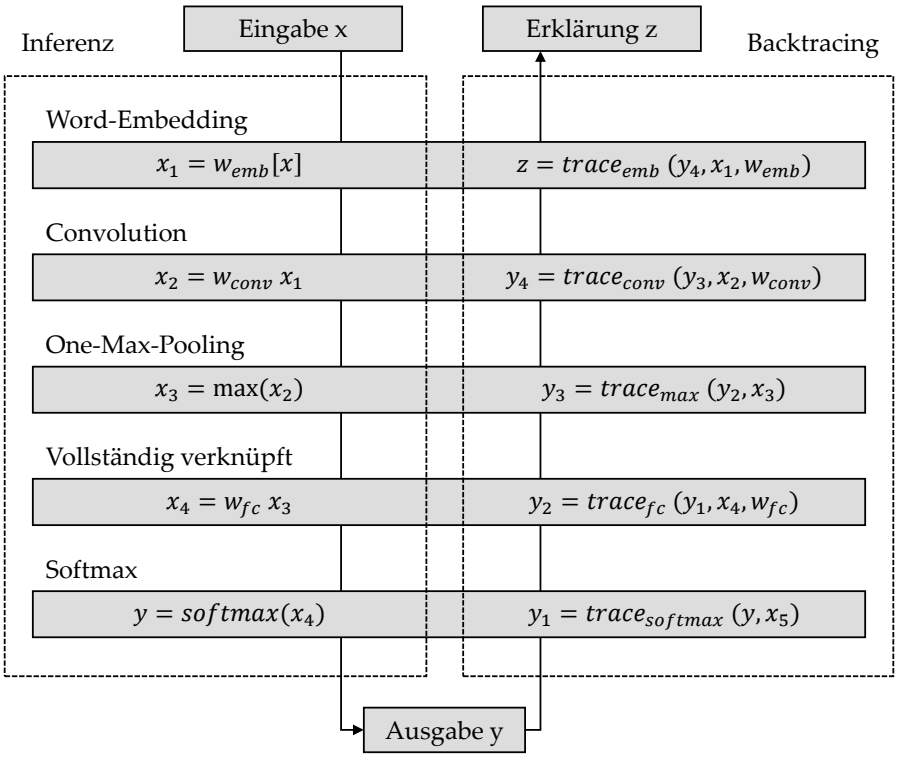


Abbildung 5.3: Nachverfolgen von Klassifikationsentscheidungen

5.2.1 Nachverfolgen von Netzwerkentscheidungen

Der Ansatz zur Nachverfolgung von Netzwerkentscheidungen basiert auf der Rückverfolgung der Berechnungen, die im Netzwerk zu einem bestimmten Ergebnis geführt haben. Für jede Operation, die das Netzwerk zur Berechnung des Ergebnisses durchführt, wird eine Umkehroperation definiert, welche den Einfluss der Eingabewerte der jeweiligen Operation auf das Klassifikationsergebnis berechnet.

Abbildung 5.3 veranschaulicht den Ansatz. In der linken Hälfte sind vereinfacht die Operationen dargestellt, die das Netzwerk zur Berechnung eines Klassifikationsergebnisses durchführt. Jede Operation verwendet das Ergebnis der vorangegangenen Operation, um so Schritt für Schritt das Klassifikationsergebnis zu berechnen. Analog dazu berechnen die Umkehroperationen unter Verwendung der Zwischenergebnisse y_n und der Gewichte der Operationen Schritt für Schritt die Erklärung z für eine Eingabe.

Definition der Operationen des CNNs

Bevor die Umkehroperationen definiert werden, werden die Operationen des verwendeten neuronalen Netzwerks präzise definiert. Diese Operationen berechnen schrittweise zu einem Eingabedokument die Klassifikationsentscheidung. Der Aufbau des Netzwerks ist dazu in Abbildung 5.4 dargestellt. In dieser Abbildung werden ebenfalls Namen für verschiedene Netzwerkeigenschaften definiert, die in den folgenden Berechnungen häufig verwendet werden:

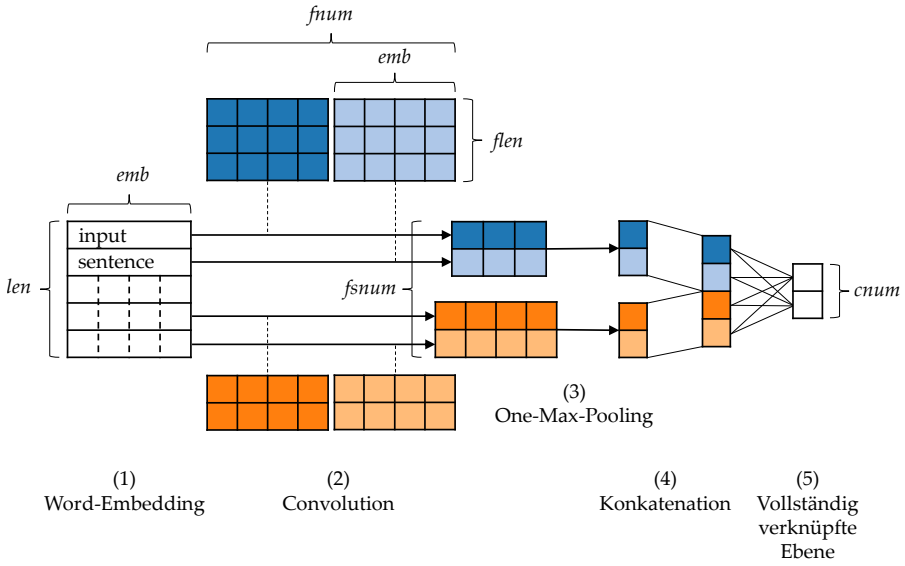


Abbildung 5.4: Aufbau CNNs für Textklassifizierung (detailliert) [57]

- $\mathtt{len}$: Die Länge des Eingabesatzes, nachdem dieser auf eine fest definierte Länge reduziert/erweitert wurde.
- $\mathtt{emb}$: Die Anzahl der Dimensionen des verwendeten Word-Embedding-Modells.
- $\mathtt{flen}$: Die Länge der Filter in einer Filtermenge. Eine Filtermenge enthält alle Filter einer Länge. In der Abbildung sind zwei Filtermengen dargestellt (blau und orange).
- $\mathtt{fnum}$: Die Anzahl der Filter in einer Filtermenge. Diese Anzahl ist für alle Filtermengen gleich.
- $\mathtt{fsnum}$: Die Anzahl der Filtermengen im Netzwerk. Das abgebildete Netzwerk besitzt zwei Filtermengen mit jeweils zwei Filtern der Länge drei (blau) und zwei (orange).
- $\mathtt{cnum}$: Die Anzahl der Ausgabeklassen.

Definition: Word-Embedding

Das Eingabedokument ist ein Vektor $\mathtt{input} \in \mathbb{N}^{\mathtt{len}}$, wobei jedes Element des Vektors ein Wort ist und durch einen Index im Wörterbuch repräsentiert wird. Als Erstes wird die Word-Embedding-Operation durchgeführt. Dabei wird jeder Wortindex durch den Embedding-Vektor $v \in \mathbb{R}^{\mathtt{emb}}$ des zugehörigen Worts aus dem Word-Embedding-Modell ersetzt. Es entsteht eine Dokumentmatrix $s \in \mathbb{R}^{\mathtt{len},\mathtt{emb}}$.

Definition: Convolution

Daraufhin wendet die Convolution-Operation die Filter auf die Dokumentmatrix an. Ein Filter ist eine Matrix $f \in \mathbb{R}^{\mathtt{flen},\mathtt{emb}}$ und enthält trainierbare Gewichte. Eine Filtermenge enthält $\mathtt{fnum}$ Filter einer Länge $\mathtt{flen}$. Eine Filtermenge ist ein Tensor $\mathtt{fs} \in \mathbb{R}^{\mathtt{fnum},\mathtt{flen},\mathtt{emb}}$. Mehrere Filtermengen können verwendet werden, um Filter unterschiedlicher Längen zu verwenden.

Filter werden auf die Dokumentmatrix angewendet, indem sie wie ein Fenster über die Dokumentmatrix geschoben werden. An jeder möglichen Position des Filters wird ein einziger Wert wie folgt berechnet, insgesamt entsteht so für jede Filtermenge fs eine Matrix $y^{(1,fs)} \in \mathbb{R}^{fnum,len+1-flen}$:

$$y_{i,j}^{(1,fs)} = \sigma\left(\left(\sum_{k=1}^{flen}\sum_{l=1}^{emb} fs_{i,k,l} \cdot s_{j+k-1,l}\right) + b_i\right) \tag{5.3}$$

σ ist die verwendete Aktivierungsfunktion, beispielsweise Sigmoid, tanh oder ReLU. Unterschiedliche Indizierungen durch j resultieren in unterschiedlichen Elementen der Dokumentmatrix s, die mit den Gewichten des Filters multipliziert werden. b_i sind weitere Gewichte (Bias) für jeden Filter, die ebenfalls trainierbar sind.

Definition: One-Max-Pooling

One-Max-Pooling wählt aus den Ergebnissen der Filteranwendungen der Convolution-Operation für jeden Filter jeweils diejenige Anwendung mit der größten Aktivierung aus. Daraus entsteht für jede Filtermenge ein Vektor $y^{(2,fs)} \in \mathbb{R}^{fnum}$:

$$y_i^{(2,fs)} = \max\left(y_{i,1}^{(1)}, y_{i,2}^{(1)}, ..., y_{i,len+1-flen}^{(1)}\right) \tag{5.4}$$

Definition: Konkatenation

Die Ergebnisse des Poolings aller Filtermengen werden nun aneinandergehängt. Bei $fsnum$ Filtermengen entsteht so ein Vektor $y^{(3)} \in \mathbb{R}^{fsnum \cdot fnum}$:

$$y^{(3)} = y^{(2,1)} \| y^{(2,2)} \| ... \| y^{(2,fsnum)} \tag{5.5}$$

Definition: Vollständig verknüpfte Ebene

Dieser Feature-Vektor zeigt an, welche der von den Filtern erlernten Wortmuster im Eingabesatz erkannt wurden. Mit einer vollständig verknüpften Ebene werden diese Features mit den Ausgabeklassen assoziiert. Bei $cnum$ Klassen entsteht so ein Vektor $y^{(4)} \in \mathbb{R}^{cnum}$:

$$y_i^{(4)} = \sigma\left(\left(\sum_{j=1}^{fsnum \cdot fnum} w_{j,i} \cdot y_j^{(3)}\right) + b_i\right) \tag{5.6}$$

In dieser Gleichung sind $w \in \mathbb{R}^{fsnum \cdot fnum, cnum}$ und $b \in \mathbb{R}^{cnum}$ Matrizen trainierbarer Gewichte. w enthält dabei die Gewichte, welche die Features mit den Ausgabeklassen assoziieren. σ ist wie bei Convolution eine beliebige Aktivierungsfunktion.

Definition: Softmax

Als letztes werden die Ergebnisse mit der softmax-Funktion in Wahrscheinlichkeiten umgerechnet.

$$y_c = \frac{e^{y_c^{(4)}}}{\sum_{i=1}^{cnum} e^{y_i^{(4)}}} \tag{5.7}$$

5.2.2 Berechnen von Einflussmatrizen

Der Ansatz zur Nachverfolgung von Netzwerkentscheidungen basiert auf der Berechnung von Einflussmatrizen. Eine Einflussmatrix IM zeigt für eine Referenzmatrix oder einen Referenzvektor r an, wie groß der Einfluss jedes einzelnen Werts von r auf ein bestimmtes Klassifikationsergebnis ist. Einflussmatrizen besitzen jeweils eine Dimension mehr als die zugehörige Referenzmatrix (bzw. der zugehörige Referenzvektor), wobei die Größe der letzten Dimension der Anzahl der Klassen $cnum$ des Netzwerks und die Größen der anderen Dimensionen denen der Referenzmatrix r entsprechen. Ein Wert $x \in IM$ mit $x > 0$ zeigt an, dass das zugehörige Element in r für die Klassifikation der Eingabe als der zu x zugehörigen Klasse spricht, wohingegen ein Wert $x < 0$ gegen die Klassifikation dieser Klasse spricht. Je größer x relativ zu den anderen Werten in IM, desto größer der Einfluss auf das Klassifikationsergebnis.

Beispielsweise ist die für den Eingabevektor $input \in \mathbb{N}^{len}$ zugehörige Einflussmatrix eine Matrix $IM \in \mathbb{R}^{len,cnum}$. Für die Eingabe „Das System muss funktionieren .“ und das Klassifikationsergebnis $(0,98, 0,02)$ (entspricht der Klasse Anforderung) kann die Einflussmatrix für den Eingabevektor wie folgt aussehen:

$$IM = \begin{pmatrix} 0,6 & 3,4 & 7,2 & 0,2 & 0,3 \\ 0,1 & 0 & -2 & 0,1 & 0,1 \end{pmatrix} \tag{5.8}$$

Diese Matrix zeigt an, dass insbesondere das Vorkommen der Wörter „System muss“ für die Klassifikation als Anforderung spricht. Insbesondere spricht das Vorkommen des Worts „muss“ gegen (-2) die Klassifikation als Information. Die anderen Wörter haben keinen relevanten Einfluss auf das Klassifikationsergebnis.

Drei Arten von Einflussmatrizen

Es wird zwischen drei Arten von Einflussmatrizen unterschieden:

DOKUMENTEINFLUSSMATRIX (DIM): Diese Matrix zeigt für die Eingabe in einem Netzwerk an, wie groß der Einfluss aller Teile der Eingabe auf alle Ausgabeklassen ist. Das Beispiel in Formel 5.8 ist eine Dokumenteinflussmatrix.

ZWISCHENEINFLUSSMATRIX (IIM): Während der Berechnung der Dokumenteinflussmatrix entstehen wie auch bei der Berechnung des Klassifikationsergebnisses Zwischenergebnisse. Diese Zwischenergebnisse sind ebenfalls Einflussmatrizen. Durch die Umkehrung jeder Netzwerkoperation wird auf Basis der letzten Zwischeneinflussmatrix eine neue Einflussmatrix berechnet, die den Einfluss der Eingabe der gerade umgekehrten Operation auf die Klassifikation anzeigt. Die Dokumenteinflussmatrix ist demnach lediglich eine spezielle Bezeichnung für diejenige Zwischeneinflussmatrix, die nach der Umkehrung aller Netzwerkoperationen entsteht.

STARTEINFLUSSMATRIX (SIM): Die Starteinflussmatrix ist eine speziell vorbereitete Einflussmatrix, die als Eingabe für die Umkehrung der letzten Netzwerkoperation dient.

Definition der Starteinflussmatrix

Im Folgenden wird beginnend mit der Netzwerkausgabe für jede Netzwerkoperation die zugehörige Umkehroperation definiert. Zuvor wird auf Basis der Netzwerkausgabe y die Starteinflussmatrix wie folgt definiert. Die Starteinflussmatrix ist eine Diagonalmatrix $SIM \in \mathbb{R}^{cnum,cnum}$ mit

$$SIM = \begin{pmatrix} y_1 & 0 & \dots & 0 \\ 0 & y_2 & & 0 \\ \vdots & & \ddots & 0 \\ 0 & 0 & 0 & y_{cnum} \end{pmatrix} \tag{5.9}$$

Diese Einflussmatrix spiegelt einen trivialen Zusammenhang zwischen der Netzwerkausgabe und der Klassifikationsentscheidung wider: das n-te Element der Netzwerkausgabe y hat genau y_n Einfluss (entspricht der Wahrscheinlichkeit für die Klasse n) auf die Klassifikation der Netzwerkeingabe als Klasse n und keinen Einfluss auf sämtliche anderen Klassen. Bei der Netzwerkausgabe $(0,98, 0,02)$ aus dem obigen Beispiel lautet die Starteinflussmatrix wie folgt:

$$SIM = \begin{pmatrix} 0,98 & 0 \\ 0 & 0,02 \end{pmatrix} \tag{5.10}$$

Umkehrung: Softmax

Die Softmax-Operation ist die letzte Operation des Netzwerks und rechnet die Ausgabe der vorangegangenen Operation in Wahrscheinlichkeiten um. Gemäß Formel 5.7 hängt jedes Element der Ausgabe y von jeweils allen Elementen der Eingabe $y^{(4)}$ ab. Der Divisor der Softmax-Berechnung ist jedoch für alle Elemente der Ausgabe y identisch, somit lässt sich folgende Aussage über die Softmax-Operation treffen: Ist ein Element y_m der Ausgabe größer als ein weiteres Element y_n der Ausgabe, so ist auch das entsprechende Element der Eingabe $y_m^{(4)}$ größer als das entsprechende Element $y_n^{(4)}$, da $e^x > e^y \Rightarrow x > y$ gilt. Anders formuliert: Besitzt ein Element nach der Anwendung der Softmax-Operation großen Einfluss auf die Ausgabe, so besitzt es auch vor der Anwendung der Softmax-Operation großen Einfluss auf die Ausgabe. Demnach verändert die Softmax-Operation den Einfluss der Eingabe auf das Klassifikationsergebnis nicht und die Umkehroperation für Softmax ist somit die Identitätsfunktion:

$$\begin{aligned} &trace_{softmax} : \mathbb{R}^{(n,cnum)} \to \mathbb{R}^{(n,cnum)} \\ &iim \mapsto iim \end{aligned} \tag{5.11}$$

Umkehrung: Vollständig verknüpfte Ebene

Die Formel 5.6 zur Berechnung der Ausgabe der vollständig verknüpften Ebene kann zum besseren Verständnis der Operation auch wie folgt umgeschrieben werden:

$$y_i^{(4)} = w_{1,i} \cdot y_1^{(3)} + w_{2,i} \cdot y_2^{(3)} + ... + w_{m,i} \cdot y_m^{(3)} + b_i \tag{5.12}$$

Das Ergebnis dieser Formel ist für die korrekte Klasse viel größer als das Ergebnis für die anderen Klassen. Demnach müssen in der Berechnung des Ergebnisses für die korrekte Klasse mehr $w_{j,i} \cdot y_j^{(3)}$ Summanden mit hohen Werten vorhanden sein als in den Berechnungen der Ergebnisse für die anderen Klassen. Ein Summand mit einem hohen Wert hat demnach großen positiven Einfluss auf das Ergebnis, während ein Summand mit negativem Wert negativen Einfluss auf das Ergebnis besitzt. Weiterhin kann jeder dieser Summanden genau einem Eingabewert zugeordnet werden. Die Summanden mit den größten Zwischenergebnissen zeigen also genau die Eingabewerte an, die am meisten Einfluss auf das Ergebnis haben.

Die Umkehroperation für vollständig verknüpfte Ebenen lautet auf Basis dieser Informationen wie folgt:

$$\begin{aligned} &trace_{fcl} : \mathbb{R}^{(onum,cnum)} \to \mathbb{R}^{(inum,cnum)} \\ &iim \mapsto [m_{i,c}] \end{aligned} \tag{5.13}$$

wobei m wie folgt definiert ist:

$$m_{i,c} = \sum_{j=1}^{onum} iim_{j,c} \cdot w_{i,j} \cdot y_i^{(3)} \tag{5.14}$$

Diese Formel wird wie folgt gelesen: Der Einfluss einer bestimmten Eingabe i auf eine Klasse c ist definiert als die Summe aller für diese Klasse relevanten Summanden $w_{j,i} \cdot y_j^{(3)}$, multipliziert mit dem Einfluss der jeweils zugehörigen Ausgabe auf diese Klasse. Durch diese Multiplikation wird Einfluss propagiert: Wenn ein Ausgabewert keinen Einfluss ($= 0$) auf eine bestimmte Klasse hat, so haben zugehörige Eingabewerte ebenfalls keinen Einfluss auf das Ergebnis. Gleiches gilt für stark positiven oder negativen Einfluss.

Umkehrung: Konkatenation

Bei der Konkatenation der Ausgabevektoren der One-Max-Pooling-Operation werden mehrere Feature-Vektoren zu einem Vektor verbunden. Analog dazu wird zur Umkehrung dieser Operation die Zwischeneinflussmatrix in mehrere Matrizen zerlegt, deren Dimensionen denen der Eingabevektoren der Konkatenationsoperation entsprechen:

$$\begin{aligned} &trace_{concat} : \mathbb{R}^{(n,cnum)} \times \mathbb{N} \to \mathbb{R}^{(fnum,cnum)} \\ &iim, i \mapsto [m_{j,c}] \end{aligned} \tag{5.15}$$

wobei m wie folgt definiert ist:

$$m_{j,c} = iim_{(i-1)\cdot fnum+j,c} \tag{5.16}$$

Der Index $i \in [1, fsnum]$ gibt an, welcher Teil der Zwischeneinflussmatrix iim zurückgegeben wird. Alle weiteren Umkehroperationen werden jeweils auf alle dieser Zwischeneinflussmatrizen angewendet. Die Zwischenergebnisse werden später wieder kombiniert.

Umkehrung: One-Max-Pooling

Die One-Max-Pooling-Operation reduziert einen Vektor auf einen einzigen Wert, wobei aus dem Vektor der jeweils größte Wert ausgewählt wird. Daher hat auch nur genau dieser eine ausgewählte Wert Einfluss auf das Klassifikationsergebnis. Änderungen an den anderen Werten verändern das Klassifikationsergebnis nicht, solange sie nicht größer als der jeweils größte Wert sind. Die Umkehroperation für One-Max-Pooling lautet daher wie folgt:

$$\begin{aligned} &trace_{1max} : \mathbb{R}^{(fnum,cnum)} \to \mathbb{R}^{(fnum,len+1-flen,cnum)} \\ &iim \mapsto [m_{i,j,c}] \end{aligned} \tag{5.17}$$

wobei m wie folgt definiert ist:

$$m_{i,j,c} = \begin{cases} iim_{j,c} & y_{i,j}^{(1)} = y_j^{(2)} \\ 0 & \text{sonst} \end{cases} \tag{5.18}$$

Diese Operation reicht den Einfluss der Matrix iim an genau derjenigen Eingabe der One-Max-Pooling-Operation weiter, an der die größte Aktivierung anliegt. Alle anderen Eingaben der One-Max-Pooling-Operation haben keine Auswirkungen auf das Klassifikationsergebnis und damit den Einfluss 0.

Umkehrung: Convolution

Die Umkehrung der Convolution-Ebene folgt den gleichen Prinzipien wie die Umkehrung von vollständig verknüpften Ebenen. Auch in Formel 5.3 wird das Ergebnis aus mehreren Summanden $fs \cdot s$ berechnet, wobei Summanden mit positiven Werten für und Summanden mit negativen Werten gegen die Aktivierung einer bestimmten Ausgabe der Convolution-Operation sprechen. Die Formel zur Umkehrung der Convolution-Operation ist daher ähnlich zu Formel 5.14, lediglich die Indizierung der Filter und der Dokumentmatrix ist aufwändiger:

$$\begin{aligned} &trace_{conv} : \mathbb{R}^{(fnum,len+1-flen,cnum)} \to \mathbb{R}^{(len,emb,cnum)} \\ &iim \mapsto [m_{i,j,c}] \end{aligned} \tag{5.19}$$

wobei m wie folgt definiert ist:

$$m_{i,j,c} = \sum_{k=1}^{fnum} \sum_{l=1}^{len+1-flen} iim_{k,l,c} \cdot s_{i,j} \cdot \begin{cases} fs_{k,i+1-l,j} & i+1-l \in [1, flen] \\ 0 & \text{sonst} \end{cases} \tag{5.20}$$

Die Fallunterscheidung verhindert lediglich, dass die Filtermenge fs nicht mit ungültigen Indizes indiziert wird.

Umkehrung: Embedding

Die Word-Embedding-Operation ersetzt die Wortindizes im Eingabevektor x durch jeweils einen Wortvektor. Die Elemente der Wortvektoren hängen damit direkt vom Wortindex ab. Wenn viele Elemente eines Wortvektors großen Einfluss auf das Klassifikationsergebnis haben, muss auch das zugehörige Wort großen Einfluss auf das Klassifikationsergebnis haben. Hat ein Wortvektor w_1 in Summe mehr Einfluss als ein anderer Wortvektor w_2, so hat das zu w_1 zugehörige Wort mehr Einfluss auf das Klassifikationsergebnis als das zu w_2 zugehörige Wort. Die Umkehrung der Word-Embedding-Operation wird demnach als Summe über alle Elemente jedes Wortvektors definiert:

$$\begin{aligned} &trace_{emb} : \mathbb{R}^{(len,emb,cnum)} \to \mathbb{R}^{(len,cnum)} \\ &iim \mapsto [m_{i,c}] \end{aligned} \tag{5.21}$$

wobei m wie folgt definiert ist:

$$m_{i,c} = \sum_{j=1}^{emb} iim_{i,j,c} \tag{5.22}$$

Einzelne Elemente der Wortvektoren können unterschiedlich großen Einfluss auf das Klassifikationsergebnis haben, der Einfluss eines Worts auf das Klassifikationsergebnis wird allerdings durch lediglich einen Wert repräsentiert. An dieser Stelle wird ein Kompromiss auf Kosten der Vollständigkeit der Erklärung und zu Gunsten der Interpretierbarkeit eingegangen. Es ist möglich, dass ein Wortvektor Elemente mit negativem Einfluss und gleichermaßen viele Elemente mit positivem Einfluss besitzt. Das zugehörige Wort hat damit kontroversen Einfluss auf das Klassifikationsergebnis, durch die obige Formel verschwindet dieser Einfluss allerdings vollständig.

Kombination der Einflussmatrizen mehrerer Convolution-Operationen

Die bei der Wort-Embedding-Operation entstehende Dokumentmatrix s wird als Eingabe für mehrere Convolution-Operationen verwendet. Die Umkehrung der Word-Embedding-Operation vereint dementsprechend alle Zwischeneinflussmatrizen, die durch die Umkehrung der Convolution-Operationen entstanden sind. Auf diese Zwischeneinflussmatrizen wird zunächst die Operation $trace_{emb}$ jeweils individuell angewendet. Die entstehenden Einflussmatrizen werden elementweise summiert. Dabei greift die gleiche Argumentation, die auch

bereits für die Addition des Einflusses der Elemente der Wortvektoren angewendet wurde. Dies ist ein Kompromiss zwischen Interpretierbarkeit und Vollständigkeit. An dieser Stelle geht die Information verloren, ob zwei benachbarte und einflussreiche Wörter individuell oder nur in Kombination miteinander zum Klassifikationsergebnis geführt haben.

Als Ergebnis entsteht eine Matrix $\mathrm{DIM} \in \mathbb{R}^{(len,cnum)}$. Dies ist die Dokumenteinflussmatrix.

Anwenden des Ansatzes bei anderen Netzwerkarchitekturen

Der Ansatz ist grundsätzlich auch auf andere Netzwerkarchitekturen übertragbar. Für jede Operation im Netzwerk muss wie für die hier verwendete CNN-Architektur gezeigt eine geeignete Umkehroperation definiert werden, die entsprechend der Operation eine Zwischeneinflussmatrix in eine weitere Zwischeneinflussmatrix umrechnet. Die Domänen und Co-Domänen dieser Operationen müssen genauso wie die der Netzwerkoperationen zusammenpassen.

Um beispielsweise die Klassifikationsentscheidungen des in Abschnitt 5.1 vorgestellten Klassifikationsmodells mit Kontextunterstützung zu erklären, müssen die vorgestellten Umkehroperationen entsprechend der Netzwerkarchitektur kombiniert werden. In dieser Netzwerkarchitektur werden vier parallele Convolution-Operationen und One-Max-Pooling-Operationen eingesetzt, deren Ergebnisse konkateniert werden. Diese Konkatenation muss entsprechend der Formeln 5.15 und 5.16 umgekehrt werden und die vier parallel durchgeführten Operationen analog der weiter vorgestellten Umkehroperationen umgekehrt werden. Für jede der vier Komponenten entsteht so eine eigene visualisierbare Dokumenteinflussmatrix. An diesen Matrizen kann daraufhin auch abgelesen werden, welche Komponente den größten Einfluss auf ein Klassifikationsergebnis besitzt.

5.2.3 Visualisierung von Klassifikationseinflüssen

Der vorgestellte Ansatz zur Berechnung von Dokumenteinflussmatrizen kann nun wie folgt verwendet werden, um Visualisierungen des Einflusses auf die Klassifikationsentscheidungen eines CNNs zu erstellen.

1. Ein CNN wird instanziiert und auf einem beliebigen Datensatz trainiert.
2. Ein Beispiel wird mit dem Netzwerk klassifiziert.
3. Das Ergebnis der Klassifikation, die Zwischenergebnisse der Klassifikation und die Netzwerkgewichte werden verwendet, um mit dem oben vorgestellten Algorithmus eine Dokumenteinflussmatrix zu berechnen.
4. Die Dokumenteinflussmatrix wird normalisiert, sodass jeder Wert der Matrix zwischen 0 und 1 liegt.

DATENSATZ	KLASSEN	GRÖSSE	GENAUIGKEIT
Questions	6	5.452	83,70%
Movie Reviews	2	10.662	73,29%
Anforderungen	2	69.567	85,10%

Tabelle 5.5: Datensätze und Ergebnisse des Klassifikators

5. Für jede Klasse wird eine Farbe gewählt.

6. Für jedes Wort im Eingabebeispiel wird jeweils die Klasse mit dem größten Einfluss gewählt. Der normalisierte Einfluss bestimmt die Stärke der zur Klasse zugehörigen Farbe, mit der das Wort hinterlegt wird.

5.2.4 Evaluation der Visualisierungen

Die Eignung des vorgestellten Ansatzes zur Visualisierung von Einflüssen auf die Klassifikation und die Genauigkeit der Visualisierungen wurde mit drei Methoden auf drei Datensätzen getestet. Die Methoden und die Ergebnisse werden im Folgenden vorgestellt.

Datensätze der Evaluation

Alle Tests basieren auf den folgenden Datensätzen:

QUESTIONS: Der Questions-Datensatz[2] enthält Fragen, die nach bestimmten Entitäten (Personen, Orte, Daten, Zahlen, Beschreibungen und Objekte) fragen. Ein Klassifikator muss bestimmen, nach welchem Typ von Information eine Frage fragt. Die meisten dieser Beispiele sind sehr einfach anhand bestimmter Fragewörter klassifizierbar, daher eignet sich dieser Datensatz besonders gut für eine erste Verifizierung der Funktionstauglichkeit des Ansatzes.

MOVIE REVIEWS: Dieser Datensatz[3] enthält kurze Reviews von Spielfilmen, die entweder positiv oder negativ sind. Hierbei handelt es sich um ein klassisches Sentiment-Analysis-Problem.

ANFORDERUNGEN UND INFORMATIONEN: Dies ist der deutsche Datensatz, der in Kapitel 4 erstellt wurde.

Methoden zur Evaluation der Visualisierungen

Für jeden Datensatz wurde zuerst ein Klassifikationsmodell erstellt und auf dem Datensatz trainiert. Die Trainingsergebnisse sind in Tabelle 5.5 grob dargestellt. Auf diesen Modellen wurden daraufhin die folgenden Evaluationen durchgeführt:

1. Zuerst wurde die generelle Tauglichkeit des Ansatzes anhand von wenigen manuell ausgewählten Stichproben untersucht. Die-

2 http://cogcomp.cs.illinois.edu/Data/QA/QC/

3 https://www.cs.cornell.edu/people/pabo/movie-review-data/

se Evaluation untersucht die generelle Fähigkeit des Ansatzes, sinnvolle Visualisierungen zu erzeugen.

2. In einem zweiten Test wurden für alle Klassen der Datensätze auf Basis der Dokumenteinflussmatrizen die jeweils einflussreichsten Wörter ermittelt. Diese Wortlisten sollten für jede Klasse jeweils repräsentative Wörter enthalten.

3. In einem dritten Test wurde empirisch ermittelt, ob und wie weit die vom Ansatz hervorgehobenen Wörter mit den von Anwendern tatsächlich als relevant erachteten Wörtern übereinstimmen.

Evaluation: Stichproben

Als erstes wurden während der Entwicklung des Ansatzes stichprobenartig Visualisierungen einzelner Beispiele aus allen Datensätzen erstellt. Dazu wurden alle Datensätze in eine Trainings- und Testmenge zerlegt und jeweils ein CNN auf den Trainingsmengen trainiert. Aus den Testmengen wurden daraufhin Beispiele ausgewählt, welche die Eigenschaften der Visualisierung besonders gut repräsentieren. Tabelle 5.6 zeigt diese Beispiele.

Die ersten beiden Beispiele für den Questions-Datensatz sind Fragen, die nach einer Beschreibung, bzw. einer Zahl fragen. Dies hat das CNN richtig erkannt und die Visualisierung zeigt, dass jeweils die Umgebung der Fragewörter („Why did“, „How long does“) ausschlaggebend für die Klassifikation ist. Dies ergibt Sinn, da es für die Klassifikation irrelevant ist, nach genau welche Beschreibung oder welcher Zahl gefragt wird. Lediglich die Tatsache, dass nach einer Beschreibung bzw. Zahl gefragt wird, ist relevant.

Das CNN konnte das dritte Beispiel nicht eindeutig klassifizieren und hat für zwei Klassen ähnlich große Wahrscheinlichkeiten ausgegeben. Die Visualisierung zeigt in diesem Fall an, welche Teile des Satzes für die Klasse *Person* und welche für die Klasse *Entity* sprechen. Der Algorithmus verbindet die Wortgruppe „associated with“ mit der Klasse *Entity*, da 90% der Beispiele, in denen diese Wortgruppe verwendet wurde, dieser Klasse zugeordnet sind. Insofern ist diese Visualisierung nicht intuitiv, aber konsistent mit der zugrundeliegenden Datenmenge.

Das CNN hat das erste Beispiel des Movie Review Datensatzes richtig klassifiziert. Die erste hervorgehobene Wortgruppe enthält die Wörter „enjoyable film“, die intuitiv der positiven Klasse zugeordnet werden können. Die zweite Wortgruppe „its own“ hat keinen offensichtlichen Zusammenhang mit der Klasse. Eine mögliche Ursache für die Hervorhebung dieser Wortgruppe ist, dass insbesondere die Wortgruppe „its own right“ häufig und fast ausschließlich in positiven Beispielen vorkommt.

Die beiden negativen Beispiele wurden vom Klassifikator ebenfalls richtig klassifiziert. Das erste der beiden Beispiele ist kurz und gehört

DATENSATZ: QUESTIONS	
Klassen	Description Numeric Human Entity
Description	Why did the chicken cross the road?
Numeric	How long does it take sunlight to reach Earth?
Human	Which of the following people is not associated with Andy Warhol?
DATENSATZ: MOVIE REVIEWS	
Klassen	Positive Negative
Positive	Both a successful adaptation and an enjoyable film in its own right.
Negative	. . . unbearably lame.
Negative	Just a bunch of good actors flailing around in a caper that's neither original nor terribly funny.
Positive	It's a minor comedy that tries to balance sweetness with coarseness, while it paints a sad picture of the singles scene .
DATENSATZ: ANFORDERUNGEN	
Klassen	Anforderung Information
Anforderung	Der Wechsel in den Zustand Aktiv muss stets in den initialen Zustand Aktiv führen.
Anforderung	Wenn die Funktion aktiv ist, darf das System die Eskalationsstufe o nicht aktivieren.
Information	Diese Objekte werden nicht als Objekte angesehen, sondern werden in einem separaten Kapitel behandelt.
Information	In den folgenden Abschnitten werden die über das System ABC gesendeten und empfangenen Signale spezifiziert.
Information	Damit wird zum Beispiel bei Änderung der Helligkeit die Lichtverteilung nicht verändert.
Information	Die Spannungswerte gelten für alle vier Türen.

Tabelle 5.6: Beispielvisualisierungen aus unterschiedlichen Datensätzen

wegen des Worts „lame“ eindeutig zur negativen Klasse. Der Visualisierungsansatz hat dies korrekt hervorgehoben. Das zweite Beispiel wurde insbesondere wegen der negativen Wortgruppe „neither x nor terribly funny“ der negativen Klasse zugeordnet. Dies ist besonders bemerkenswert, da das Wort „funny“ alleinstehend eher ein Indiz für die positive Klasse ist.

Das letzte Beispiel konnte nicht eindeutig einer Klasse zugeordnet werden. Dementsprechend wurden sowohl die Teile des Satzes hervorgehoben, die für die positive Klasse sprechen als auch die Teile, die für die negative Klasse sprechen. Dieses Review ist sehr neutral geschrieben und kein Teil des Satzes erlaubt eine eindeutige Zuordnung zu einer der beiden Klassen. Die Hervorhebung von „tries to balance“ ist schlüssig, da diese Formulierung oftmals zur Beschreibung von Sachverhalten benutzt wird, die ein Film zu erreichen versucht, es aber nicht schafft.

Auf dem Anforderungsdatensatz markiert der Ansatz in Anforderungen sehr häufig Modalverben als ausschlaggebendes Kriterium für die Klassifikation. Auch werden meist mehr die grammatikalischen Strukturen der Anforderungen hervorgehoben und weniger die in den jeweiligen Anforderungen konkret spezifizierten Sachverhalte. In den gezeigten Anforderungen sind beispielsweise die Wörter „muss“ und „darf“ die einflussreichsten Wörter. Die Wortgruppe „Wenn die Funktion aktiv ist“ ist für die Klassifikation unwichtig, da für die Klassifikation in Anforderungen und Informationen nicht relevant ist, welche exakte Funktion durch die Anforderung spezifiziert wird.

In Informationen werden ebenfalls für Informationen typische Wörter hervorgehoben. Die erste Information ist ein Verweis auf ein anderes Kapitel. In der zweiten Information wird der Inhalt eines bestimmten Abschnitts des Dokuments beschrieben. Eine weitere Information verdeutlicht die Relevanz einer Anforderung durch ein Beispiel. In diesen Anforderungen werden Wörter und Wortgruppen wie „Kapitel“, „den folgenden Abschnitten“ und „Beispiel“ hervorgehoben. In der zweiten Information ist ebenfalls „das System ABC“ hervorgehoben. Diese Wortgruppe wird häufig in Verbindung mit Abschnittsbeschreibungen verwendet. Die letzte Information verdeutlicht einen Zusammenhang, der aus den betreffenden Anforderungen (hier nicht dargestellt) möglicherweise nicht sofort ersichtlich ist. „Gelten“ scheint häufig für Informationen dieser Art verwendet zu werden. Die Hervorhebung des Worts „Türen“ ist möglicherweise ein Indiz auf Overfitting.

Evaluation: Einflussreichste Wörter

Der erste Test hat für ausgewählte Beispiele gezeigt, dass der Ansatz in der Lage ist, sinnvoll die einflussreichsten Wörter hervorzuheben. Im zweiten Test wird versucht, diese Beobachtung auf jeweils der gesamten Testmenge aller Datensätze durchzuführen. Dazu wurde für jedes Beispiel in den Testmengen der Datensätze die Dokumenteinflussmatrix berechnet.

KLASSE	EINFLUSSREICHSTE WÖRTER
Abbreviation	stand, does, abbreviation, mean, is, for, an, number, beer, fame, term
Description	how, what, is, do, are, the, does, why, a, origin, did, of, difference, mean
Entity	what, the, name, was, is, of, a, fear, are, for, does, did, to, do, color
Human	who, what, was, name, the, of, and, actor, company, school, tv
Location	where, country, city, can, capital, the, was, state, are, what, does, in, of
Numeric	how, many, when, did, does, was, year, long, the, do, average, is, of

Tabelle 5.7: Questions-Datensatz: Einflussreichste Wörter

KLASSE	EINFLUSSREICHSTE WÖRTER
Positive	that, is, of, a, has, film, us, with, best, makes, an, worth, performances, it's, fun, who, good, documentary, funny
Negative	but, too, and, more, bad, no, so, movie, in, or, doesn't, just, only, about, the, there's, i, are, isn't

Tabelle 5.8: Movie-Reviews-Datensatz: Einflussreichste Wörter

Für jedes in der Testmenge vorkommende Wort wurde der Einfluss, welchen alle Vorkommen dieses Worts auf das Klassifikationsergebnis haben, summiert und zur Normalisierung durch die Anzahl der Vorkommen dividiert. Für jede Klasse wurde das Wörterbuch absteigend nach dem normalisierten Gesamteinfluss sortiert. Hat ein Wort durchschnittlich größeren Einfluss auf das Klassifikationsergebnis, so erscheint es weiter vorne in der Liste als ein Wort mit geringerem Einfluss. Die einflussreichsten Wörter für eine Klasse sollten demnach Wörter sein, die charakteristisch für diese Klasse sind.

Tabelle 5.7 zeigt die Ergebnisse für den Questions-Datensatz. Die meisten Klassen können anhand bestimmter Fragewörter und Pronomen sehr einfach erkannt werden. Fragen, die „how many“ oder „when“ enthalten erwarten meist eine Zahl als Antwort, Fragen mit „who was“ erwarten eine Person als Antwort und das Schlüsselwort „where“ deutet auf Fragen nach Orten hin. Die Tabelle zeigt, dass dies tatsächlich die jeweils am häufigsten hervorgehobenen Wörter sind.

Die Ergebnisse auf dem Movie-Reviews-Datensatz sind in Tabelle 5.8 dargestellt. Die Liste der Wörter, die großen Einfluss auf die positive Klasse haben, enthält erwartete Wörter wie „best“, „worth“,

KLASSE	EINFLUSSREICHSTE WÖRTER
Anforderung	muss, darf, soll, wenn, sein, werden, o, das, von, signalisieren
Information	wird, beschrieben, kann, Kapitel, Anforderungen, Komponente, sind, Abteilung, Beispiel, Telefon, Abbildung, Rollo, sich, Beschreibung, definiert, siehe, folgenden, diesem, bezeichnet, E-Mail, verwendet, zum, im, da, stellt, Komponenten, dient, auch, gibt

Tabelle 5.9: Anforderungs-Datensatz: Einflussreichste Wörter

„fun“, „good“ und „funny“. Allerdings sind auch Wörter enthalten, die nicht direkt mit dieser Klasse in Verbindung gesetzt werden („that“, „is“, „of“, ...). Diese Wörter werden scheinbar besonders häufig in Wortgruppen mit positiver Stimmung verwendet. Das Wort „but“ wird oft in negativen Reviews benutzt („The plot was okay, *but* ...“) und ist das Wort mit dem größten Einfluss auf diese Klasse. Weiterhin sind viele negative Wörter mit negativer Stimmung enthalten („bad“, „doesn't“, „no“, „isn't“, ...).

Tabelle 5.9 zeigt die einflussreichsten Wörter der Klassen Anforderung und Information. Wie auch durch die vorherige Auswertung der Stichproben bereits angedeutet entscheiden besonders Modalverben wie „muss“, „darf“ und „soll“ über die Klassifikation als Anforderung. „Wenn“ wird häufig in Anforderungen verwendet, da mit diesem Wort bedingte Eigenschaften formuliert werden. Die Zahl „o“ steht aufgrund der Vorverarbeitungsschritte stellvertretend für alle Ziffern und Zahlen. Werden Zahlen in Spezifikationsobjekten verwendet, so geschieht dies meist zur Spezifikation von Spannungen, Dimensionen, Zeitabständen, etc.

Die Liste der einflussreichsten Wörter für die Klasse Information ist deutlich länger. Während Anforderungen anhand weniger Wörter identifiziert werden können, sind Informationen deutlich diversifizierter. Die Liste enthält Wörter für Beschreibungen („wird“, „beschrieben“, „Beschreibung“, „bezeichnet“), Beispiele, Abbildungen, Verweise („Komponente(n)“, „Kapitel“, „siehe“, „folgenden“) und generell schwächere Wörter, die nicht in Anforderungen verwendet werden sollen („kann“, „sind“, „dient“, „auch“, „gibt“). „Abteilung“, „Telefon“ und „E-Mail“ sind in dieser Liste aufgeführt, da ein häufig vorkommendes Informationsobjekt die Nennung eines Ansprechpartners ist. „Rollo“ wird ebenfalls häufig in Informationsobjekten verwendet und ist daher ein Indiz auf Overfitting des Klassifikators auf den als Information klassifizierten Objekten einer bestimmten Spezifikation.

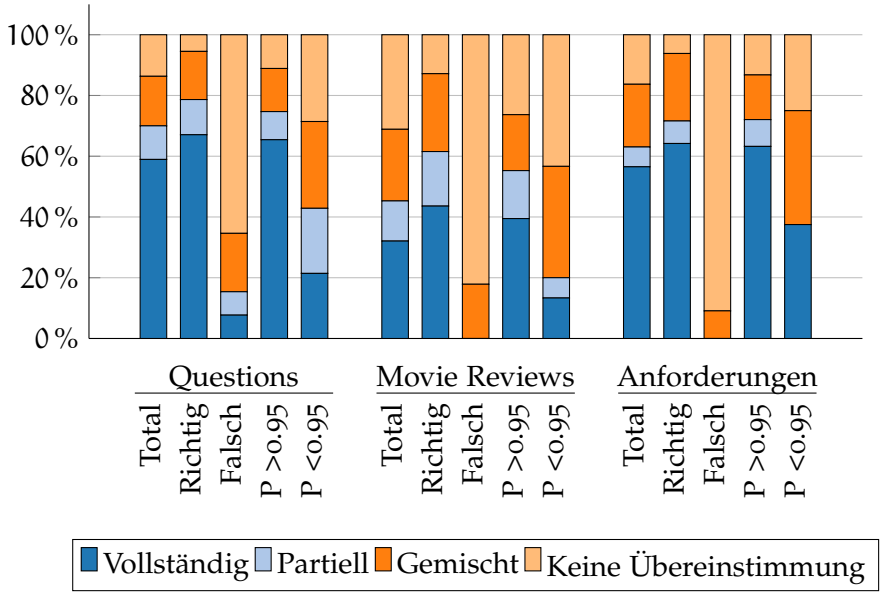


Abbildung 5.5: Auswertung der Übereinstimmung der generierten Visualisierungen

Evaluation: Empirische Übereinstimmungstests

Weiterhin wurde eine empirische Studie durchgeführt, um die Qualität der Visualisierung detaillierter zu bewerten. Für jedes Beispiel in den Testmengen der Datensätze wurde zuerst manuell bestimmt, welche Wörter in diesem Beispiel ausschlaggebend für die Klassifizierung sein sollten. Da die Testmengen der Datensätze sehr groß sind, wurden die Testmengen für diese Auswertungen auf 200 zufällig ausgewählte Objekte reduziert.

Ausschlaggebende Wörter sind Wörter, die für die Klassifikation eines Beispiels in eine bestimmte Klasse sprechen. Für den Movie-Reviews-Datensatz sind dies beispielsweise Wörter, die entweder eine positive oder negative Stimmung äußern. Im Anforderungsdatensatz sind bestimmte Modalverben besonders ausschlaggebend für die Klasse Anforderung.

Daraufhin wurde die Erwartung mit der tatsächlichen Visualisierung verglichen. Jedes Beispiel wurde dann anhand des Grades der Übereinstimmung in eine der folgenden vier Kategorien eingeordnet:

VOLLSTÄNDIGE ÜBEREINSTIMMUNG: Der Ansatz hat in der Visualisierung genau die erwarteten Wörter hervorgehoben. Keine weiteren Wörter wurden hervorgehoben.

PARTIELLE ÜBEREINSTIMMUNG: Der Ansatz hat einige, aber nicht alle der erwarteten Wörter in der Visualisierung hervorgehoben. Keine weiteren, unerwarteten Wörter wurden hervorgehoben.

GEMISCHTE ÜBEREINSTIMMUNG: Der Ansatz hat einige oder alle der erwarteten Wörter hervorgehoben. Zusätzlich wurden allerdings weitere Wörter hervorgehoben, die nicht für die Klassifikation ausschlaggebend sein sollten.

KEINE ÜBEREINSTIMMUNG: In der Visualisierung wurden kein einziges der erwarteten Wörter hervorgehoben.

Die Ergebnisse sind in Abbildung 5.5 gesammelt dargestellt. Für jeden Datensatz zeigt die Grafik fünf Balken. Der erste Balken zeigt die Ergebnisse aller bewerteten Beispiele. Um die Ergebnisse besser auswerten zu können, sind die Ergebnisse zudem getrennt nach richtig und falsch klassifizierten Beispielen und getrennt nach Wahrscheinlichkeit (größer und kleiner 95%) dargestellt.

Im Questions-Datensatz sind bei etwa 59% aller Beispiele exakt alle relevanten Wörter hervorgehoben. Bei weiteren 11% der betrachteten Beispiele sind nur einige relevante Wörter hervorgehoben, bei den restlichen 30% sind ebenfalls irrelevante Wörter markiert. Die Trennung der Ergebnisse in richtig und falsch klassifizierte Beispiele zeigt, dass der Ansatz grundsätzlich nicht in der Lage ist, bei falsch klassifizierten Beispielen sinnvolle Visualisierungen zu erzeugen. Andererseits befinden sich unter den richtig klassifizierten Beispielen nur wenige Beispiele, bei denen nicht mindestens ein relevantes Wort hervorgehoben wurde. Die Trennung nach Wahrscheinlichkeit der Korrektheit zeigt, dass bei Beispielen mit höherer Wahrscheinlichkeit bessere Visualisierungen entstehen.

Die Ergebnisse auf dem Movie-Reviews-Datensatz sind erheblich schlechter. Nur bei 32% der Beispiele wurde eine korrekte Visualisierung erzeugt. Bei insgesamt 68% der Beispiele wurden zumindest einige relevante Wörter hervorgehoben. Bei 55% der Beispiele sind zudem nicht relevante Wörter hervorgehoben. Die im Vergleich zum Questions-Datensatz insgesamt schlechteren Ergebnisse sind auf die schlechtere Genauigkeit des Klassifikators auf dem Movie-Reviews-Datensatz zurückzuführen (siehe Tabelle 5.5). Von den schlechten Ergebnissen abgesehen spiegeln sich die auf dem Questions-Datensatz gemachten Beobachtungen an den nach Klassifikationsergebnis und Klassifikationswahrscheinlichkeit getrennten Beispielen auch auf diesem Datensatz wider.

Auf dem Anforderungs-Datensatz wurden 57% aller Beispiele gemäß den Erwartungen hervorgehoben. Bei 16% der Beispiele sind die Visualisierungen in keiner Weise hilfreich. Bei den restlichen 27% sind entweder nicht alle relevanten Wörter oder zusätzliche, nicht relevante Wörter hervorgehoben. Auch hier wiederholen sich die auf den anderen beiden Datensätzen gemachten Beobachtungen.

Zusammenfassung der Evaluationsergebnisse

Auf Basis der drei durchgeführten Untersuchengen können folgenden Aussagen über den Visualisierungsansatz getroffen werden:

- Der Ansatz ist grundsätzlich in der Lage, relevante Wörter hervorzuheben.
- Bei Beispielen, die nicht eindeutig einer bestimmten Klasse angehören, werden Wörter für mehrere mögliche Klassenkandidaten hervorgehoben. Diese Eigenschaft ist insbesondere beim Einsatz

in einem Softwarewerkzeug relevant, da der Anwender so bei Beispielen mit nicht eindeutiger Klassenzugehörigkeit besser entscheiden kann, welche Teile des Beispiels ggf. zu ändern sind.

- Die Genauigkeit der Visualisierungen ist grundsätzlich von der Klassifikationsgenauigkeit abhängig. Bessere Ergebnisse des Klassifikators auf den Testdatenmengen führen zu besseren Visualisierungen.
- Die vom Klassifikator nach der Klassifikation eines Beispiels ausgegebene Wahrscheinlichkeit kann für die Einschätzung der Güte der Visualisierung verwendet werden: Bei Beispielen mit höherer Wahrscheinlichkeit sind die Visualisierungen in der Regel besser.
- Durch die Erstellung von Listen einflussreicher Wörter können CNN-Modelle auf Overfitting überprüft werden. Das Modell zur Klassifikation von Anforderungen und Informationen hat beispielsweise erlernt, bestimmte, sehr spezifische Objekte als Information zu klassifizieren (Ansprechpartner und Objekte, die „Rollo“ enthalten). Diese Erkenntnisse können zur weiteren Verbesserung des Datensatzes und des Klassifikators verwendet werden.

6 EVALUATION

In diesem Kapitel wird der in den vorangegangenen Kapiteln vorgestellte Ansatz zur Klassifikation von Anforderungen und Informationen evaluiert. Die Evaluation unterteilt sich in mehrere Schritte. In Abschnitt 6.1 wird zuerst eine ausführliche statistische Auswertung durchgeführt. Dazu wird Random-Search verwendet, um die optimalen Hyperparameter für den Klassifikator zu identifizieren. In Abschnitt 6.2 werden die Ergebnisse eines empirischen Experiments vorgestellt. Zwei Gruppen führten ein Review einer Spezifikation durch, wobei eine Gruppe das Review manuell durchführte und die andere Gruppe durch automatische Klassifikation unterstützt wurde. Im Rahmen dieses Experiments wurde der Ansatz ebenfalls in ein Werkzeug integriert. In Abschnitt 6.3 wird ein weiteres Experiment durchgeführt, um Recall und Precision des Klassifikators zu optimieren.

6.1 STATISTISCHE AUSWERTUNG: RANDOM-SEARCH

Zur Optimierung der Hyperparameter des Klassifikators wurde Random-Search [17] eingesetzt. Bei dieser Optimierungstechnik werden wiederholt zufällig gewählte Netzwerkkonfigurationen trainiert und getestet. Mit den Ergebnissen kann daraufhin der Einfluss jedes Hyperparameters auf die Klassifikationsgenauigkeit untersucht, die optimale Netzwerkkonfiguration ermittelt und damit die bestmögliche Genauigkeit des Netzwerks bestimmt werden.

Parameterbereiche für Random-Search

Die Hyperparameter bewegen sich in den in Tabelle 6.1 angegebenen Bereichen. Bei den beiden Hyperparametern Komponenten und Filterlängen wird jeweils eine beliebige Kombination der möglichen Werte ausgewählt. Es kann sowohl nur das Objekt selbst oder auch Objekt, Vorgänger und Nachfolger zur Klassifikation verwendet werden. Analog dazu sind Netzwerke mit ausschließlich Filtern der Länge 1 als auch Filtern der Längen 2, 3 und 4 möglich.

Insgesamt wurden 1.818 Versuche durchgeführt, bei denen jeweils das Netzwerk gemäß der zufällig gewählten Konfiguration instanziiert und anschließend mittels zehnfacher Kreuzvalidierung auf dem englischen Datensatz trainiert wurde. Für die Durchführung benötigte eine Workstation mit zwei Grafikkarten des Typs NVIDIA Quadro

HYPERPARAMETER	WERTEBEREICH
Embedding-Größe	8 - 128
Trainierbare Embeddings	Ja / Nein
Maximale Satzlänge	10 - 100
Komponenten	Beliebige Kombination aus Objekt, Elternobjekt, Vorgänger und Nachfolger
Filterlängen	1 - 4 und beliebige Kombinationen daraus
Anzahl Filter je Länge	1 - 32

Tabelle 6.1: Bereiche der Hyperparameter für Random-Search

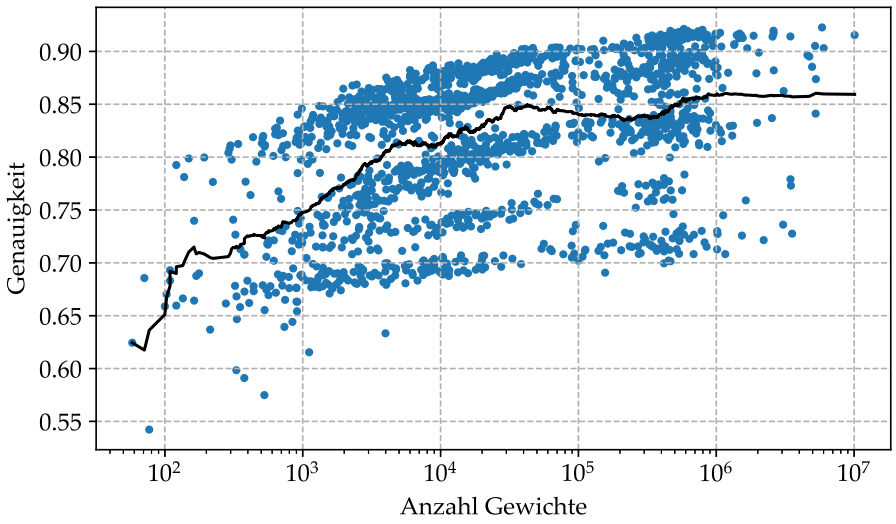


Abbildung 6.1: Random-Search: Anzahl Gewichte

P5000 insgesamt sechs Tage. Die Ergebnisse werden nun anhand jedes Hyperparameters separat analysiert.

Anzahl Gewichte

Abbildung 6.1 zeigt die Genauigkeit des Klassifikators auf dem Testdatensatz in Abhängigkeit der Anzahl der trainierbaren Gewichte. In dieser Grafik ist erkennbar, dass die Klassifikationsgenauigkeit grundsätzlich mit der Anzahl der Gewichte steigt. Weiterhin ist die Bandbreite an möglichen Ergebnissen bei Netzwerken vergleichbarer Größe sehr groß. Viele Gewichte führen demnach nicht notwendigerweise zu guten Ergebnissen.

Verwenden von Kontextinformationen

In der Grafik sind Cluster zu erkennen: Besonders im Bereich zwischen 10^4 und 10^5 Gewichten sammeln sich die Ergebnisse in fünf Gruppen um die Genauigkeiten 0,70, 0,74, 0,80, 0,87 und 0,89. Ursache für diese Gruppierung sind unterschiedliche Kombinationen der für die Klassifikation verwendeten Komponenten. In Abbildung 6.2 sind alle Ergebnisse, bei denen sowohl das Objekt selbst als auch das Elternobjekt verwendet wurde dunkelblau gekennzeichnet. Hellblau

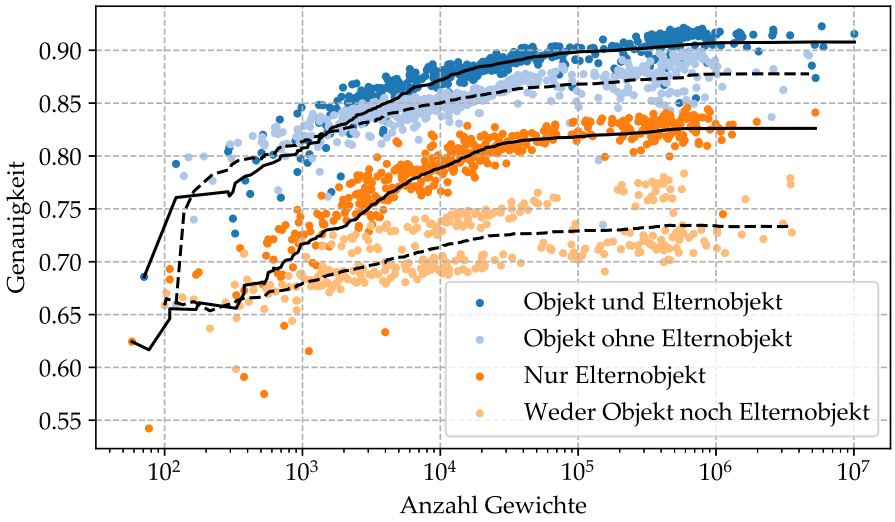


Abbildung 6.2: Random-Search: Kontextkomponenten

sind alle Ergebnisse markiert, bei denen das Objekt selbst, aber nicht das Elternobjekt verwendet wurde. Die Ergebnisse mit und ohne Elternobjekt ohne Verwendung des eigentlichen Objekttexts sind orange gekennzeichnet.

An dieser Grafik spiegeln sich die bereits in Kapitel 5.1.2 gemachten Beobachtungen wider. Durch die Verwendung von Kontextinformationen (insb. das Elternobjekt) können deutlich bessere Ergebnisse erzielt werden. Die Klassifikation eines Objekts ausschließlich mit Kontextinformationen führt erwartungsgemäß zu deutlich schlechteren Ergebnissen. Erwähnenswert ist dennoch, dass Genauigkeiten von 80% und besser allein auf Basis des Kapitels, in dem ein Objekt erscheint, erreicht werden können.

Abbildung 6.3 zeigt den Einfluss von Vorgänger und Nachfolger auf das Klassifikationsergebnis: Insgesamt ist nicht erkennbar, dass diese Komponenten zu einer positiven Veränderung der Ergebnisse beitragen. Aufgrund der gemachten Beobachtungen bezüglich des Einflusses des Kontextes auf die Genauigkeit werden in den folgenden Analysen nur diejenigen Ergebnisse betrachtet, bei denen sowohl das Objekt selbst als auch das Elternobjekt verwendet wurde.

Trainierbare Word-Embeddings

In Abbildung 6.4 ist der Einfluss trainierbarer Word-Embeddings abgebildet: Durch die Anpassung der Embeddings an das konkrete Klassifikationsproblem können deutlich bessere Ergebnisse erzielt werden. Die maximal erreichbare Genauigkeit steigt dadurch von 90,3% auf 92,3%.

Filteranzahl

Abbildung 6.5 zeigt den Einfluss der kombinierten Gesamtzahl der Filter auf die Genauigkeit. Obwohl die maximale Filteranzahl auf 32 begrenzt wurde, sind in dieser Grafik größere Werte möglich, da sich die Filteranzahl auf jeweils jede Filterlänge bezieht. Ein Netzwerk mit 32 Filtern der Längen 1, 2, 3 und 4 besitzt insgesamt 128 Filter. Durch die Steigerung der Filteranzahl erlernt das Netzwerk mehr

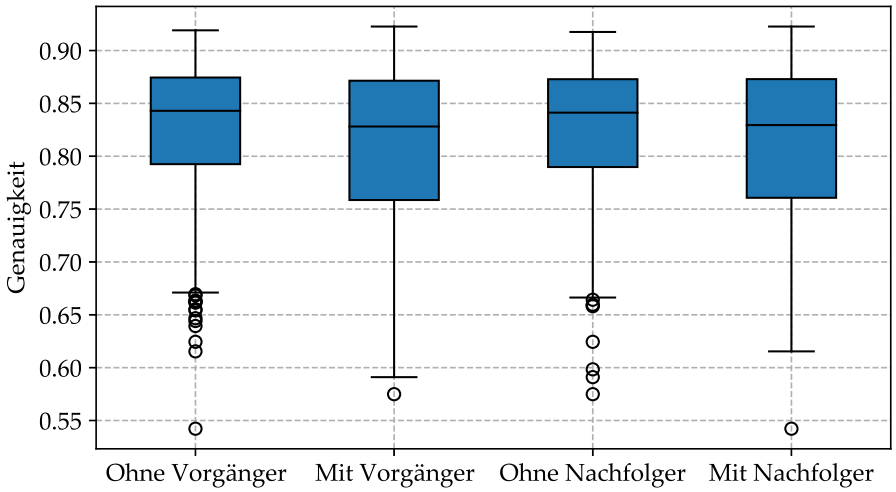


Abbildung 6.3: Random-Search: Vorgänger- und Nachfolgerobjekte

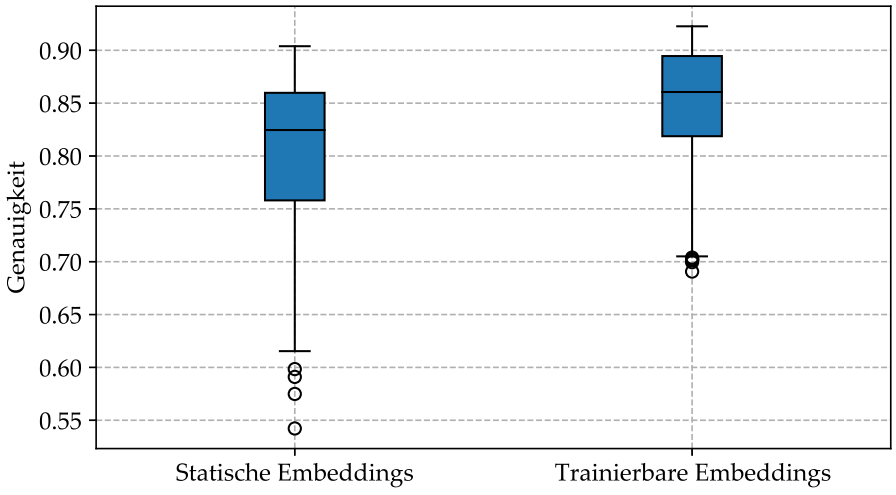


Abbildung 6.4: Random-Search: Trainierbare Embeddings

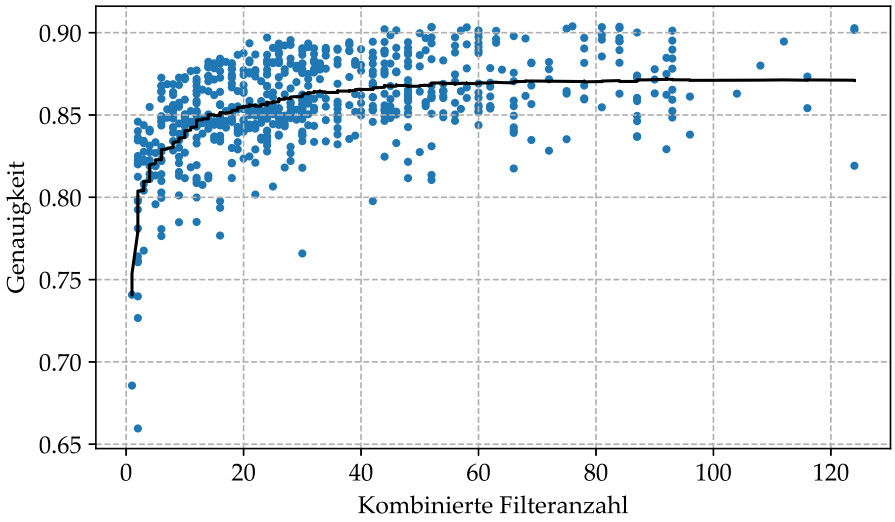


Abbildung 6.5: Random-Search: Gesamtanzahl der Filter

unterschiedliche Muster, um Informationen und Anforderungen zu unterscheiden. Insgesamt ist beobachtbar, das mit steigender Filteranzahl die Klassifikationsgenauigkeit steigt. Ab einer Filteranzahl von 40 werden die besten Ergebnisse erreicht, mehr Filter führen nicht zu besseren Ergebnissen. Ab diesem Punkt erlernen zusätzliche Filter zunehmend redundante Muster.

Filterlänge

Die Länge der Filter hat einen großen Einfluss auf die Leistung des Klassifikators. Die Filterlänge gibt an, wie viele Wörter das vom Filter erkannte Muster umfasst. Abbildung 6.6 zeigt vier Box-Plots. Der erste Box-Plot zeigt die Ergebnisse mit maximaler Filterlänge 1. Der zweite Box-Plot zeigt Ergebnisse mit maximaler Filterlänge 2, exklusive der Datenpunkte des ersten Plots. Es ist erkennbar, dass mit steigender maximaler Filterlänge die Genauigkeit des Klassifikators steigt. Die maximal erreichbare Genauigkeit steigt durch die Erhöhung der maximalen Filterlänge von 1 auf 3 von 88% auf 91%. Die Verwendung von Filtern der Länge 4 bringt keine deutlichen Verbesserungen gegenüber der maximalen Filterlänge 3.

Satzlänge

Die Satzlänge hat keinen großen Einfluss auf die Leistung des Klassifikators (siehe Abbildung 6.7). Dies bedeutet, dass der Klassifikator bei den meisten Beispielen bereits anhand der ersten Wörter über die Klasse entscheiden kann. Ebenfalls sind ein Großteil der Beispiele im Datensatz nicht länger als 40 Wörter (siehe dazu Abbildung 4.1 auf Seite 46), eine größere Satzlänge ist damit lediglich zur Abdeckung von selten auftretenden, längeren Beispielen notwendig. Eine kürzere maximale Satzlänge hat den Vorteil, dass die Klassifikation von Beispielen deutlich schneller durchgeführt wird, da dadurch insbesondere die Anzahl der Filteranwendungen durch die Convolution-Operation während der Inferenz reduziert wird.

Größe der Word-Embeddings

Abbildung 6.8 zeigt zuletzt den Einfluss der Größe der Word-Embeddings auf die Genauigkeit des Klassifikators. Umso größer

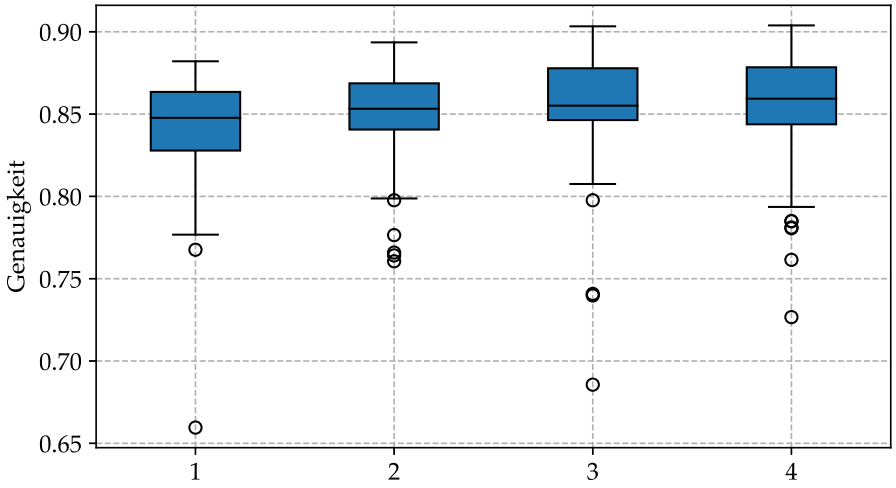


Abbildung 6.6: Random-Search: Filterlängen

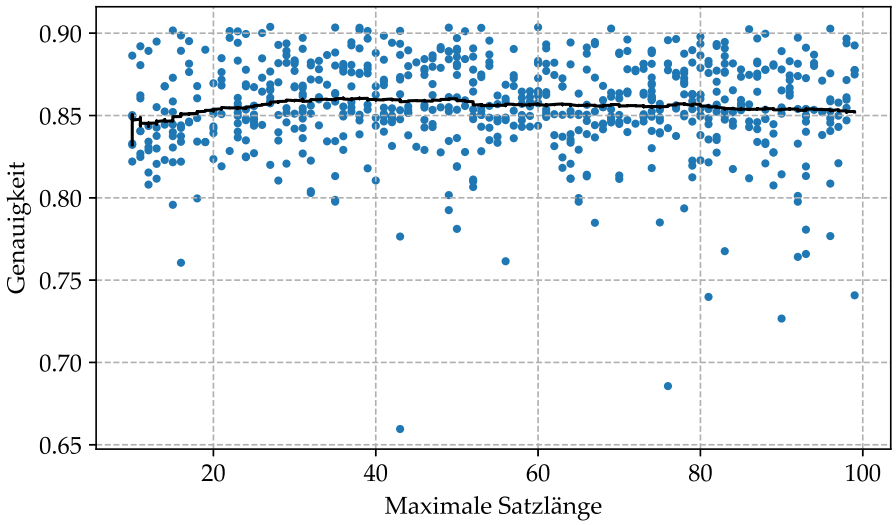


Abbildung 6.7: Random-Search: Satzlänge

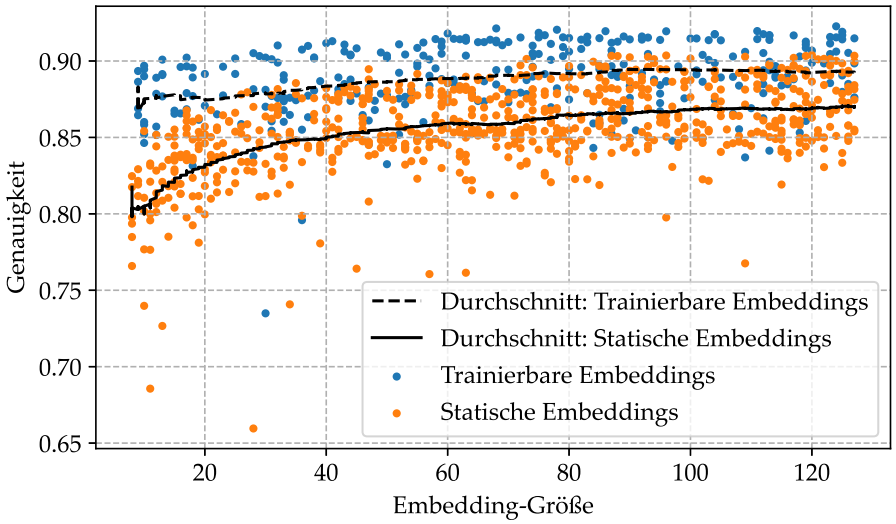


Abbildung 6.8: Random-Search: Embedding-Größe und trainierbare Embeddings

die Embeddings gewählt werden, desto mehr Möglichkeiten hat der Klassifikator, zwei verschiedene Wörter auseinander zu halten. Die Grafik zeigt, dass bei statischen Embeddings durch die Erhöhung der Größe der Embeddings deutliche Verbesserungen in der Genauigkeit erzielt werden. Die Erhöhung der Größe von 8 auf 40 führt zu einer Verbesserung der durchschnittlichen Genauigkeit um 5%. Die Genauigkeit bei der Verwendung trainierbarer Embeddings ist im Vergleich zu statischen Embeddings insgesamt besser. Allerdings ist der Vorteil durch größere Embeddings nicht so stark ausgeprägt. Auch mit kleinen Embeddings können bereits Ergebnisse mit mehr als 90% Genauigkeit erzielt werden. Dies liegt daran, dass während des Trainings die Word-Embeddings an das Klassifikationsproblem angepasst werden.

Zusammenfassung der Random-Search-Ergebnisse

Anhand der Ergebnisse können nun die Hyperparameter für das Convolutional-Neural-Network ausgewählt werden, die zu den besten Klassifikationsergebnissen führen. Bei allen Hyperparametern zeigt sich, dass eine größere Anzahl trainierbarer Gewichte generell zu besseren Ergebnissen führt. Bei allen Hyperparametern existiert allerdings auch eine Grenze, ab der jeweils keine Verbesserungen mehr erreicht werden. Die Hyperparameter werden so gewählt, dass diese Grenze nicht überschritten wird, da dies zu einem Netzwerk führt, welches in der Anwendung langsamer ist und keine bessere Klassifikation durchführen kann. Tabelle 6.2 zeigt die ausgewählten Hyperparameter.

Mit diesen Hyperparametern ist es insgesamt möglich, die folgenden Ergebnisse zu erreichen. Tabelle 6.3 zeigt die Konfusionstabelle, Tabelle 6.4 zeigt die daraus berechneten Metriken für beide Klassen. Insgesamt werden 91% aller Objekte im Testdatensatz richtig klassifiziert. Dies deckt sich mit den bereits in Abschnitt 5.1.2 (Tabelle 5.3)

HYPERPARAMETER	WERT
Embedding-Größe	40
Trainierbare Embeddings	Ja
Maximale Satzlänge	40
Komponenten	Objekt und Elternobjekt
Filterlängen	1, 2, 3
Anzahl Filter je Länge	20

Tabelle 6.2: Ausgewählte Hyperparameter

Wahrheit / Prognose	INFORMATION	ANFORDERUNG
INFORMATION	13.350	1.772
ANFORDERUNG	1.485	19.500

Tabelle 6.3: Konfusionsmatrix des CNN-Klassifikators

bei der Auswertung der Kontextklassifikation erzielten Ergebnissen. Auch dort wurde eine ähnliche Genauigkeit erreicht. Dies bestätigt auch, dass die in diesem Abschnitt verwendeten Hyperparameter sehr gut gewählt wurden.

Der Klassifikator favorisiert Anforderungen gegenüber Informationen leicht. Dies ist am gegenüber dem Recall von Informationen höheren Recall von Anforderungen erkennbar. Dies stellt allerdings kein Problem dar. Eine fälschlicherweise als Anforderung klassifizierte Information wird in späteren Entwicklungsschritten zu deutlich weniger Kosten führen als eine fälschlicherweise als Information klassifizierte und damit für viele Prozessschritte unsichtbare Anforderung. Möglicherweise ist es sogar angebracht, noch mehr Gewicht auf die Klasse Anforderung zu legen.

Die Ergebnisse sind bereits sehr vielversprechend. Welchen Effekt der Klassifikator in der Praxis haben wird, wird mit weiteren empirischen Studien untersucht.

METRIK	INFORMATION	ANFORDERUNG
Precision	0,900	0,917
Recall	0,883	0,929
F_1	0,891	0,923
Genauigkeit	0,910	

Tabelle 6.4: Metriken des CNN-Klassifikators

6.2 EXPERIMENT 1: MANUELLES UND WERKZEUGGESTÜTZTES REVIEW

Durch die im vorherigen Abschnitt gezeigten Ergebnisse wurde gezeigt, dass der Klassifikator grundsätzlich in der Lage ist, Anforderungen und Informationen zu trennen. Da die Genauigkeit des Klassifikators allerdings weit von 100% entfernt ist, ist es offensichtlich, dass dieser nicht zur automatischen Korrektur von Anforderungsdokumenten geeignet ist. Dennoch kann der Klassifikator zur Verbesserung der Qualität von Anforderungsdokumenten hinsichtlich der Klassifikation von Anforderungen und Informationen beitragen, indem er während eines Reviews die Aufmerksamkeit des RE-Experten auf diejenigen Teile des Dokuments richtet, die höchstwahrscheinlich eine Überprüfung und Korrektur benötigen.

Dazu wurde ein Softwarewerkzeug entwickelt. In einem empirischen Experiment wird daraufhin untersucht, ob die vom Werkzeug gegebenen Hinweise zu einer Verbesserung der Qualität ausgewählter Anforderungsdokumente führen, indem die Leistung zweier Gruppen von Anwendern (werkzeuggestützt und manuell) anhand von Metriken wie beispielsweise Anzahl korrigierter Fehler gegeneinander verglichen wird.

6.2.1 Werkzeugintegration

Zur Durchführung des Experiments wurde der Ansatz in einem Werkzeug implementiert. Das Werkzeug analysiert ein Anforderungsdokument und sucht in diesen nach Fehlern. Ein *Fehler* liegt vor, wenn der bei einem Objekt angegebene Objekttyp nicht mit dem wahren Objekttyp übereinstimmt. Bei Objekten, die möglicherweise falsch klassifiziert sind, schlägt das Werkzeug einen anderen Objekttyp vor. Die Umsetzung der Vorschläge liegt noch immer in der Hand des Anwenders, das Werkzeug nimmt keine automatischen Änderungen vor.

Abbildung 6.9 zeigt das Werkzeug. Es ist grundsätzlich in drei Abschnitte unterteilt. Im Zentrum wird das Anforderungsdokument angezeigt. Derzeit als Anforderung markierte Objekte werden in grüner Schrift dargestellt, Informationen in Rot. Findet das Werkzeug ein Objekt, das anders klassifiziert werden sollte, wird die gesamte Zeile deutlich sichtbar rot hinterlegt und in der Spalte „Warnings“ eine Fehlermeldung angezeigt. Weiterhin zeigt das Werkzeug auch Warnungen in gelber Farbe an. Bei Warnungen besteht eine größere Wahrscheinlichkeit auf False-Positives. Die Unterscheidung zwischen Warnungen und Fehlern wird auf Basis der vom neuronalen Netzwerk ausgegebenen Wahrscheinlichkeit und der Anzahl bekannter Wörter im Text des Objekts getroffen. Bei Objekten mit noch geringerer Wahr-

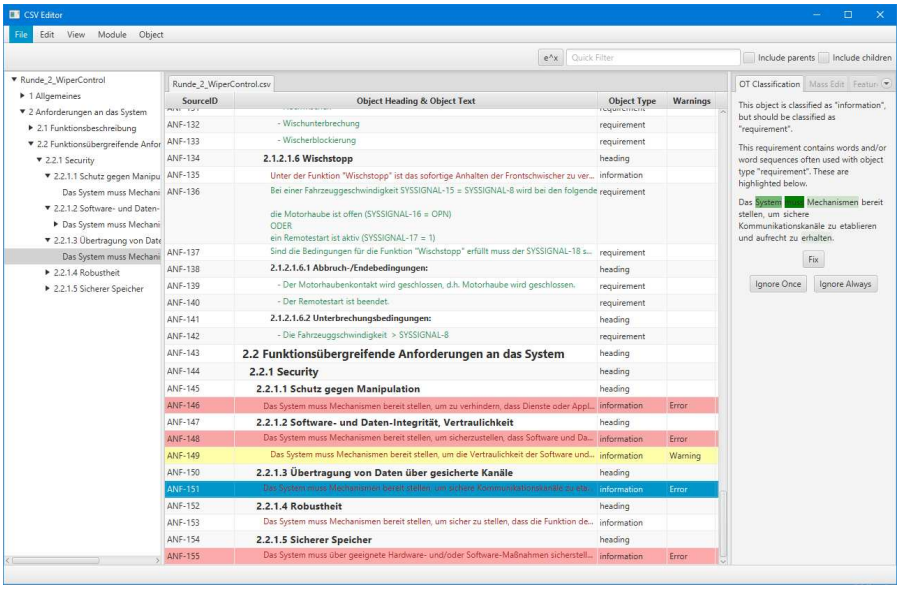


Abbildung 6.9: Werkzeug zur Unterstützung der Klassifikation von Anforderungen und Informationen

scheinlichkeit werden die Ausgaben des Klassifikators ignoriert und weder Fehler noch Warnungen angezeigt.

Der linke Teil des Werkzeugs zeigt einen Überblick über das Dokument, der die Navigation vereinfacht. Im rechten Teil werden Informationen zum aktuell ausgewählten Objekt angezeigt: Der derzeit vergebene Objekttyp eines Objekts, der durch das Werkzeug vorgeschlagene Objekttyp und die Visualisierung, welche die für die Klassifikationsentscheidung relevanten Wörter hervorhebt (siehe Kapitel 5.2). Die Schaltflächen ermöglichen es dem Anwender, die vorgeschlagene Lösung entweder anzuwenden oder zu verwerfen. Dadurch soll die Bearbeitungszeit reduziert werden. In beiden Fällen wird die Warnung entfernt. Darüber hinaus bietet das Werkzeug Funktionen zum Bearbeiten und Durchsuchen des geöffneten Dokuments.

Indem explizit Objekte mit fragwürdiger Klassifizierung hervorgehoben werden, sollten RE-Experten mehr Fehler in den Anforderungsdokumenten finden können. Möglicherweise gelingt dies durch den Einsatz des Werkzeugs sogar in kürzerer Zeit. Da das Werkzeug jedoch nicht in der Lage ist, alle Fehler zu finden, können RE-Experten durch die Fokussierung auf bestimmte Teile der Anforderungsdokumente dazu neigen, Fehler in den nicht hervorgehobenen Teilen zu übersehen.

6.2.2 Planung und Durchführung des Experiments

Zur Überprüfung der durch den Einsatz des Werkzeugs erzielbaren Verbesserungen wurde ein empirisches Experiment mit Studierenden durchgeführt. Bei der Planung, Durchführung und Auswertung des

Experiments wurden die Richtlinien von Ko, LaToza und Burnett [62] und Jedlitschka, Ciolkowski und Pfahl [53] verwendet. Das Ziel dieses Experiments ist es, herauszufinden, ob und wie der Einsatz des Werkzeugs das Finden von Fehlern hinsichtlich der Objekttypklassifikation in Anforderungsdokumenten im Vergleich zu vollständig manuellen Reviews verbessert. Da es unterschiedliche Möglichkeiten gibt, den Reviewprozess zu verbessern, wurde der Fokus auf folgende Forschungsfragen gelegt:

Definition der Forschungsfragen für das Experiment

1. *Befähigt die Benutzung des Werkzeugs den Anwender, mehr Fehler in einem Anforderungsdokument zu finden?* Dies ist das Hauptziel des Werkzeugs. Durch den Fokus auf möglicherweise falsch klassifizierte Objekte sollte ein Anwender in der Lage sein, mehr Fehler zu finden.

2. *Verringert die Benutzung des Werkzeugs die Anzahl an Fehlern, die während des Reviews entstehen?* RE-Experten machen während eines Reviews Fehler, beispielsweise können sie ein korrekt als Anforderung markiertes Objekt fälschlicherweise als Information markieren. Durch den Fokus auf falsch klassifizierte Objekte sollten RE-Experten weniger dazu neigen, nicht hervorgehobene Objekte zu ändern und dadurch Fehler zu erzeugen.

3. *Neigen Anwender dazu, Fehler zu ignorieren, die nicht durch das Werkzeug hervorgehoben wurden?* Die Anwender können dazu tendieren, Objekte ohne Hervorhebung durch das Werkzeug weniger genau zu untersuchen. Fehler in diesen Objekten können dadurch leichter übersehen werden. Dies kann bei unzureichender Genauigkeit des Werkzeugs dazu führen, dass durch den Einsatz des Werkzeugs mehr Nachteile als Vorteile entstehen.

4. *Können die Anwender mit dem Werkzeug ein Review schneller durchführen?* Ein primäres Ziel des Ansatzes ist es ebenfalls, dass RE-Experten effizienter arbeiten können. Daher wird zusätzlich untersucht, ob durch den Einsatz des Werkzeugs Zeit eingespart werden kann.

5. *Motiviert das Werkzeug Anwender dazu, Objekte umzuformulieren?* Das Werkzeug zeigt zu jedem gefundenen Fehler eine Erklärung und eine Wahrscheinlichkeit in Prozent an. Ergibt die Erklärung keinen Sinn und/oder ist der Wert sehr gering, ist dies ein Zeichen für ein Objekt, welches weder der Klasse *Anforderung* noch der Klasse *Information* eindeutig zugeordnet werden konnte. In diesen Fällen kann ein RE-Experte dazu neigen, die Formulierung des Objekts zu ändern. Ob ein solcher Zusammenhang besteht, wird ebenfalls untersucht.

Design des Experiments

Das Experiment wurde als Cross-over-Studie [113] mit zwei Durchläufen und zwei Gruppen durchgeführt. Der Aufbau des Experiments

	GRUPPE 1	GRUPPE 2
Durchlauf 1 (WWC)	Kontrollgruppe	Werkzeuggruppe
Durchlauf 2 (WL)	Werkzeuggruppe	Kontrollgruppe

Tabelle 6.5: Experiment 1: Aufbau der Cross-over-Studie

ist in Tabelle 6.5 dargestellt. Die Teilnehmer der Studie wurden zuerst in zwei Gruppen unterteilt. Daraufhin wurden zwei Durchläufe durchgeführt, wobei jeweils eine der beiden Gruppen mit dem Werkzeug arbeitete (*Werkzeuggruppe*, WG) und die andere Gruppe das Review manuell durchführte (*Kontrollgruppe*, KG). Für beide Durchläufe wurden jeweils andere Anforderungsspezifikationen verwendet.

Teilnehmer

Die Studie wurde als Teil der Lehrveranstaltung „Automotive Software Engineering" an der TU Berlin durchgeführt. Die Teilnehmer des Experiments waren Bachelor- und Master-Studierende. Ein Großteil der Studierenden studierte entweder Informatik, Technische Informatik oder Automotive Systems. Innerhalb der Lehrveranstaltung wurden ebenfalls die Themen Anforderungsmanagement und darunter auch Qualität von Anforderungen behandelt. Den Studierenden war insbesondere bewusst, welche Auswirkungen mangelhaftes Anforderungsmanagement auf spätere Entwicklungsschritte haben kann. Damit besaßen die Studierenden grundsätzlich das Wissen und die Fähigkeiten, welche zur Durchführung der Aufgaben der Studie benötigt werden.

Das Experiment wurde im Voraus angekündigt. Es wurde insbesondere betont, dass eine große Teilnehmeranzahl für eine sinnvolle Datenauswertung von großer Wichtigkeit ist. Als Motivation zur Teilnahme wurde den Studierenden Einblick in anwendungsnahes Anforderungsmanagement versprochen. Am Experiment haben 20 Studierende teilgenommen, was in etwa zwei Drittel aller in der Lehrveranstaltung eingeschriebenen Studierenden entspricht.

Aufgaben

Die Aufgaben, welche die Teilnehmer während der Studie durchführen sollten, entsprechen den Aufgaben, die auch während eines tatsächlichen Reviews von Anforderungsdokumenten durchgeführt werden. Jeder Studierende sollte die Anforderungsspezifikation lesen, verstehen, Fehler identifizieren und diese korrigieren. Folgende Fehler sollten korrigiert werden:

- Anforderungen, bei denen der Objekttyp fälschlicherweise auf Information gesetzt ist
- Informationen, bei denen der Objekttyp fälschlicherweise auf Anforderung gesetzt ist
- Ungünstig formulierte Anforderungen (z.B. mehrdeutig, fehlendes Modalverb, etc.)

Die Fehler sollten durch Anpassung des Objekttyps oder Änderung des Objekttexts korrigiert werden.

Ablauf des Experiments

Das Experiment wurde als Teil einer Lehrveranstaltung durchgeführt und musste daher innerhalb von 90 Minuten beendet werden. Die Zeit wurde in folgende vier Abschnitte unterteilt:

EINFÜHRUNG (20 MINUTEN): Die Einführung umfasste eine kurze Auffrischung des Themas Qualität von Anforderungen und eine Beschreibung des konkreten Problems mit Anforderungen und Informationen. Es wurde erklärt, wie aktuell Reviews von Anforderungsdokumenten durchgeführt werden und wie der Einsatz des Werkzeugs diese Situation verbessern soll. Danach wurden die Studierenden in zwei Gruppen eingeteilt und die Materialien für das Experiment verteilt. Das Werkzeug wurde den Teilnehmern bereits eine Woche im Voraus zur Verfügung gestellt, um Verzögerungen durch technische Probleme zu minimieren.

DURCHLAUF 1 (20 MINUTEN): Während des ersten Durchlaufs bearbeiteten die Teilnehmer die Scheibenwischer-Spezifikation je nach Gruppenzuteilung entweder mit oder ohne Werkzeug von oben nach unten und nahmen Änderungen zur Korrektur von Fehlern vor. Die Teilnehmer durften in Teams von bis zu zwei Personen arbeiten. So konnten sie bei Unsicherheiten diskutieren und gemeinsam zu einer Lösung kommen. Dies imitiert das Vorgehen in realen Reviews, da auch dort über Probleme diskutiert wird. Es wurde explizit verboten, Informationen zwischen mehreren Teams auszutauschen, da ansonsten die Unabhängigkeit der einzelnen Ergebnisse nicht mehr sichergestellt werden kann. Am Ende der Sitzung markierten die Teilnehmer die Position in den Dokumenten, bis zu der sie das Review durchgeführt hatten. Hat ein Teilnehmer das Review innerhalb der 20 Minuten beenden können, notierte diese/r stattdessen die benötigte Zeit.

DURCHLAUF 2 (30 MINUTEN): Der zweite Durchlauf wurde genauso wie auch der erste Durchlauf durchgeführt. Lediglich die Gruppen wurden für diesen Durchlauf gewechselt. Da die Fensterheber-Spezifikation etwas umfangreicher ist, wurde für diesen Durchlauf mehr Zeit eingeplant.

ZUSAMMENFASSUNG (10 MINUTEN): Nach dem zweiten Durchlauf wurden alle Ergebnisse eingesammelt. In einer kurzen Präsentation wurde gezeigt, wie diese Daten ausgewertet und welche Ergebnisse erwartet werden.

Studienmaterialien

Die Anforderungsdokumente, welche für die Studie verwendet wurden, sind aus realen Anforderungsspezifikationen abgeleitet. Das erste Dokument beschreibt ein Scheibenwischersystem und enthält eine Spezifikation der von diesem System durchzuführenden Funktionen

und der beteiligten Komponenten. Das zweite Dokument spezifiziert auf identische Weise ein Fensterhebersystem. Die Dokumente sind tabellarisch aufgebaut. Jede Zeile enthält den Text eines Objekts, einen Identifizierungsschlüssel und den zugewiesenen Objekttyp.

Diese Dokumente sind sehr groß und umfassen jeweils etwa 3.000 Objekte. Da es für die Teilnehmer unmöglich ist, innerhalb der gegebenen Zeit die Dokumente zu lesen, zu verstehen und Fehler in der Klassifizierung zu finden, wurden die Dokumente auf eine für die Studie angemessene Größe reduziert. Dazu wurden in beiden Dokumenten aus allen Funktionen nur jeweils wenige Funktionen ausgewählt, die besonders repräsentativ für das jeweilige System sind. Auch wurden Großteile der Inhalte entfernt, die aus Vorlagen stammen. Weiterhin wurden sämtliche interne Bezeichnungen konkreter Systeme, Komponenten und Signale durch Pseudonyme und sämtliche konkrete Werte (unter anderem Zeit-, Größen- und Gewichtsangaben) durch andere Werte ersetzt, dass dadurch die auf den Dokumenten durchzuführende Aufgabe nicht beeinträchtigt wird.

Für beide Dokumente wurde daraufhin ein Goldstandard erstellt. Für jedes Objekt wurde der korrekte Objekttyp ermittelt, der von dem tatsächlichen Objekttyp im Dokument abweichen kann. Dieser Goldstandard dient dem Vergleich der beiden Gruppen als Referenz.

Daraufhin wurden beide Dokumente in jeweils zwei unterschiedlichen Formaten vorbereitet: Ein CSV-Dokument zur Verwendung mit dem Werkzeug und ein Excel-Dokument, welches das Aussehen des Dokuments im Werkzeug (Schriftarten, Farben) so gut wie möglich nachstellt.

Tabelle 6.6 zeigt eine Zusammenfassung der Dokumente Scheibenwischersystem (WWC), Fensterhebersystem (WL) und Hands-Free-Access (HFA) und der Leistung des Werkzeugs auf diesen Dokumenten. Das Dokument HFA wurde für eine zweite Durchführung des Experiments verwendet (siehe weiter unten). Die Anzahl der Fehler ist die Anzahl an Objekten, bei denen die vorhandene Objekttypklassifikation vom Goldstandard abweicht. Die Genauigkeit bezieht sich auf die Fähigkeit des Klassifikators, den Goldstandard korrekt vorherzusagen.

Pilotierung

Vor der Durchführung des Experiments wurde das Experiment am Lehrstuhl erprobt. Dabei wurde das Experiment von Mitarbeitern des Lehrstuhls Automated System Engineering Technologies (ASET) so wie oben beschrieben durchgeführt. Die Ergebnisse wurden verwendet, um sicherzustellen, dass die Größe der Dokumente für die zur Verfügung stehende Zeit angemessen ist und dass das Experiment sinnvoll auswertbare Ergebnisse erzeugt.

Wiederholung des Experiments

Das Experiment wurde in der gleichen Art und Weise ein zweites Mal durchgeführt, um mehr Daten für die Auswertung zu erhalten. Das zweite Experiment wurde von einer anderen Gruppe von Studierenden auf den Dokumenten Scheibenwischer und Hands-Free-Access durchgeführt. Für die zweite Durchführung des Experiments

	WWC	WL	HFA
Anzahl Objekte	115	261	148
Anzahl Anforderungen	85	186	80
Anzahl Informationen	30	75	68
Anzahl Fehler	20	17	44
Anzahl Werkzeughinweise	24	70	38
Richtige Werkzeughinweise	12	12	32
Fehler ohne Hinweis	8	5	12
Genauigkeit	82,6%	75,8%	89,2%

Tabelle 6.6: Experiment 1: Zusammenfassung der Dokumente

wurde die Fensterheber-Spezifikation nicht verwendet, da die geringe Genauigkeit des Klassifikators auf dieser zu nicht repräsentativen Ergebnissen führte. Die folgende Auswertung fasst die Ergebnisse beider Experimente zusammen.

Datenverfügbarkeit

Die den folgenden Auswertungen zugrundeliegenden Daten sind ebenfalls online verfügbar [112].

6.2.3 Metriken zur Auswertung der Ergebnisse

Zur Auswertung der Studienergebnisse werden die im Folgenden vorgestellten Metriken verwendet. Zu jeder Forschungsfrage wird jeweils eine Metrik definiert, anhand derer die Leistung der beiden Gruppen hinsichtlich der jeweiligen Forschungsfrage verglichen wird.

FF1: Werden mehr Fehler gefunden?

Befähigt die Benutzung des Werkzeugs den Anwender, mehr Fehler in einem Anforderungsdokument zu finden? Diese Frage wird beantwortet, indem die Fehlerkorrekturrate (Defect Correction Rate, DCR) für beide Gruppen berechnet wird:

$$\text{DCR} = \frac{\text{Fehler korrigiert}}{\text{Fehler untersucht}} \tag{6.1}$$

Fehler korrigiert ist die Anzahl Fehler, die ein Teilnehmer identifiziert und korrigiert hat, *Fehler untersucht* die Anzahl an Objekten mit Fehlern, die dieser Teilnehmer während des Reviews untersucht hat. Hier wird explizit nicht die Gesamtanzahl an Fehlern im Dokument verwendet, da der Teilnehmer aufgrund der Zeitbeschränkung möglicherweise nicht alle Objekte im Dokument untersucht hat. Durch die Verwendung des Werkzeugs sollte die Fehlerkorrekturrate steigen:

$$H_1 : \text{DCR(WG)} > \text{DCR(KG)} \tag{6.2}$$

FF2: Werden weniger Fehler erzeugt?

Verringert die Benutzung des Werkzeugs die Anzahl an Fehlern, die während des Reviews entstehen? Ähnlich zu FF1 wird diese Frage beantwortet, indem die Fehlereinführrate (Defect Introduction Rate, DIR) berechnet wird:

$$\text{DIR} = \frac{\text{Fehler eingeführt}}{\text{Objekte untersucht}} \tag{6.3}$$

Fehler eingeführt ist die Anzahl an ursprünglich korrekt klassifizierten Objekten, die durch einen Teilnehmer geändert wurden und nach der Änderung vom Goldstandard abweichen. *Objekte untersucht* ist die Gesamtanzahl an Objekten, die dieser Teilnehmer untersucht hat. Da das Werkzeug korrekt klassifizierte Objekte seltener hervorhebt, sollten Anwender des Werkzeugs seltener korrekt klassifizierte Objekte ändern:

$$H_2 : \text{DIR}(\text{WG}) < \text{DIR}(\text{KG}) \tag{6.4}$$

FF3: Werden versteckte Fehler ignoriert?

Neigen Anwender dazu, Fehler zu ignorieren, die nicht durch das Werkzeug hervorgehoben wurden? Zur Evaluierung dieser Frage werden nur Objekte betrachtet, die gemäß Goldstandard fehlerhaft markiert sind und zu denen das Werkzeug keinen Hinweis gegeben hat. Die Rate nicht gefundener versteckter Fehler (Hidden Defects Miss Rate, HDMR) wird wie folgt berechnet:

$$\text{HDMR} = \frac{\text{Versteckte Fehler übersehen}}{\text{Versteckte Fehler untersucht}} \tag{6.5}$$

Versteckte Fehler übersehen ist die Anzahl an Objekten mit Fehlern und ohne Werkzeughinweis, die durch den Anwender nicht korrigiert wurden. *Versteckte Fehler untersucht* umfasst sowohl korrigierte als auch nicht korrigierte Objekte mit Fehlern ohne Werkzeughinweis, die ein Teilnehmer untersucht hat.

Obwohl während des manuellen Reviews keine Werkzeughinweise sichtbar waren, wird HDMR der Kontrollgruppe genauso wie auch für die Werkzeuggruppe berechnet. Dadurch wird die Vergleichbarkeit gewährleistet. Da das Werkzeug durch dessen Hinweise die Aufmerksamkeit des Benutzers auf Objekte mit Hinweisen verschiebt, werden Anwender des Werkzeugs wahrscheinlich weniger versteckte Fehler finden. Solange die Anzahl versteckter Fehler gering bleibt, sollte dies aber keinen großen Einfluss auf die Fehlerkorrigierrate haben.

$$H_3 : \text{HDMR}(\text{WG}) > \text{HDMR}(\text{KG}) \tag{6.6}$$

FF4: Wird Zeit eingespart?

Können die Anwender mit dem Werkzeug ein Review schneller durchführen? Diese Frage wird beantwortet, indem die durchschnittliche Zeit,

die Anwender aus den beiden Gruppen zur Bearbeitung eines Objekts benötigt haben, verglichen wird. Die Zeit pro Objekt (Time per Object, TPO) wird wie folgt berechnet:

$$\text{TPO} = \frac{\text{Verbrachte Gesamtzeit}}{\text{Objekte untersucht}} \tag{6.7}$$

Verbrachte Gesamtzeit ist die Zeit, die die Teilnehmer zur Bearbeitung des Dokuments benötigt haben. Wurde ein Teilnehmer nicht fertig, ist dies die maximale Länge des jeweiligen Durchlaufs (20, bzw. 30 Minuten). Anwender des Werkzeugs sollten insgesamt weniger Zeit zur Bearbeitung des Anforderungsdokuments benötigen, da sie durch die Werkzeughinweise lediglich einen Teil des Dokuments genauer untersuchen müssen:

$$H_4 : \text{TPO(WG)} > \text{TPO(KG)} \tag{6.8}$$

FF5: Werden Objekte umformuliert?

Motiviert das Werkzeug Anwender dazu, Objekte umzuformulieren? Die letzte Fragestellung wird durch Berechnen der Rate umformulierter Objekte (Object Rephrase Rate, ORR) beantwortet:

$$\text{ORR} = \frac{\text{Objekte umformuliert}}{\text{Objekte untersucht}} \tag{6.9}$$

Objekte umformuliert ist die Anzahl an Objekten, bei denen Anwender den Text des Objekts geändert haben. Irrelevant ist die Art der Änderung. Wenn sich der Text nach der Änderung vom Originaltext des Objekts unterscheidet, ist es ein geändertes Objekt. Der Einfluss des Werkzeugs auf diese Metrik ist unklar, möglicherweise existiert kein Zusammenhang. Ebenfalls bedeuten mehr geänderte Objekte nicht notwendigerweise einen positiven Einfluss auf die Qualität des Anforderungsdokuments. Für diese Forschungsfrage lässt sich daher keine Hypothese aufstellen; die Ergebnisse müssen individuell untersucht werden.

6.2.4 Auswertung der Ergebnisse

Für das Scheibenwischersystem wurden insgesamt 14 Reviews angefertigt, davon sieben mit Werkzeug und sieben ohne Werkzeug. Für das Fensterhebersystem wurden weniger Reviews durchgeführt (drei mit Werkzeugunterstützung und vier manuelle Reviews), da einige Studierende früher gehen mussten, ein Review keine Änderungen enthielt und ein weiteres Review verworfen werden musste, da der Teilnehmer große sprachliche Probleme hatte und das Review daher nicht durchführen konnte.

In der Wiederholung des Experiments haben die Teilnehmer für das Scheibenwischersystem weitere neun Reviews angefertigt, davon vier

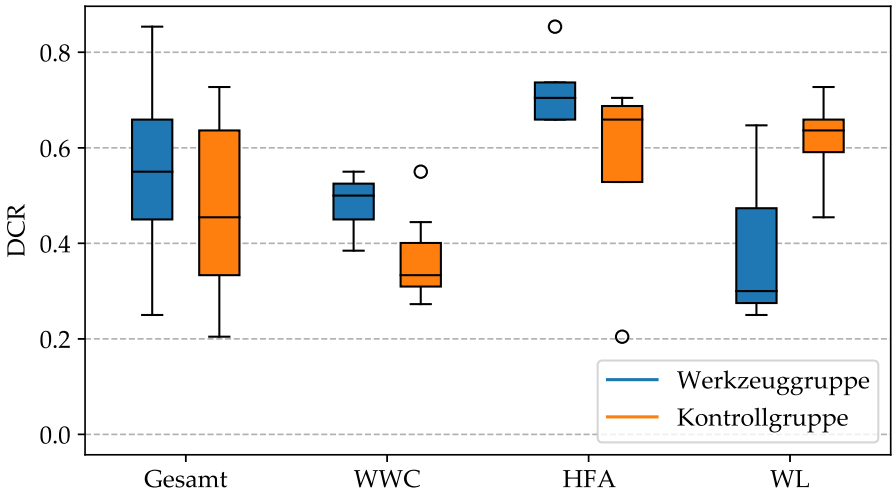


Abbildung 6.10: Experiment 1: Korrigierte Fehler

mit Werkzeug. Für das Hands-Free-Access-System stehen ebenfalls neun Reviews zur Verfügung, davon fünf mit Werkzeug.

Insgesamt wurden 4.776 Objekte untersucht, davon 2.335 mit Werkzeug (49%). Die Teilnehmer haben insgesamt 847 Objekte mit Fehlern untersucht. Für die Beantwortung der definierten Forschungsfragen wurde für jede Metrik jeweils ein Box-Plot angefertigt. Diese stellen jeweils die Ergebnisse der Kontrollgruppe und der Werkzeuggruppe zusammengefasst und separat für jedes Dokument gegenüber.

Auswertung: Rate korrigierter Fehler

Abbildung 6.10 zeigt die Rate korrigierter Fehler. Sowohl am Scheibenwischersystem als auch am Hands-Free-Access-System ist erkennbar, dass der Einsatz des Werkzeugs zu mehr korrigierten Fehlern führt. Am HFA-System ist insbesondere auffällig, dass es durch den Werkzeugeinsatz zu insgesamt konsistenteren Reviews kommt. Die Ergebnisse am Fensterhebersystem sind unter Werkzeugeinsatz deutlich schlechter. Eine Ursache dafür ist die hohe Anzahl an False-Positives, die das Werkzeug bei diesem Dokument ausgegeben hat (siehe Tabelle 6.6) und die im Vergleich zu den anderen beiden Dokumenten schlechtere Genauigkeit des Klassifikators auf dem HFA-System. Insgesamt bewirkt der Werkzeugeinsatz dennoch, dass der Anteil gefundener Fehler steigt, wodurch Hypothese H_1 bestätigt wird.

Auswertung: Rate eingeführter Fehler

Abbildung 6.11 zeigt, wie viele neue Fehler durch das Review entstanden sind. Alle drei Dokumenten bestätigen Hypothese H_2. Insgesamt entstehen bei der Verwendung des Werkzeugs im Mittel etwa nur halb so viele neue Fehler als bei manuellen Reviews. Die Ursache dafür muss die Verschiebung des Fokus des Anwenders auf wahrscheinlich falsch gekennzeichnete Objekte sein. So werden seltener richtig markierte Objekte geändert.

Auswertung: Übersehene versteckte Fehler

Bei allen drei Dokumenten steigt wie in Abbildung 6.12 erkennbar der Anteil übersehener versteckter Fehler an. Dies deckt sich mit Hypothese H_3. Durch den Werkzeugeinsatz werden im Mittel 89%

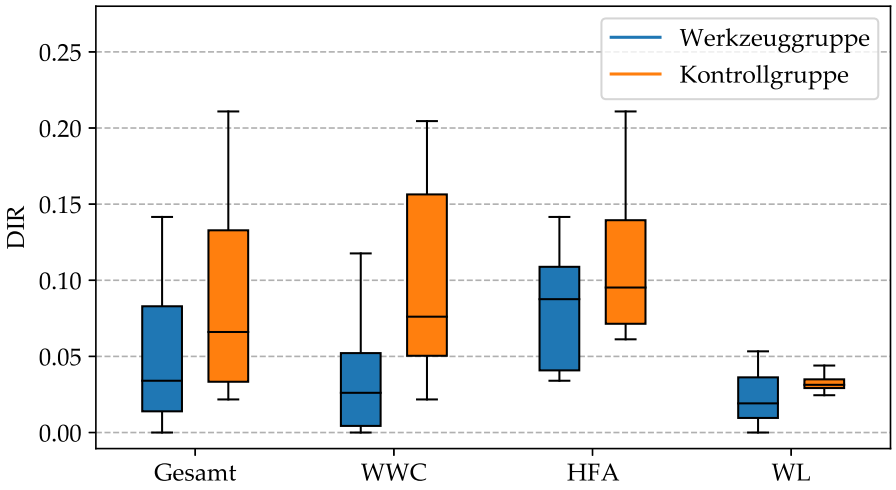


Abbildung 6.11: Experiment 1: Eingeführte Fehler

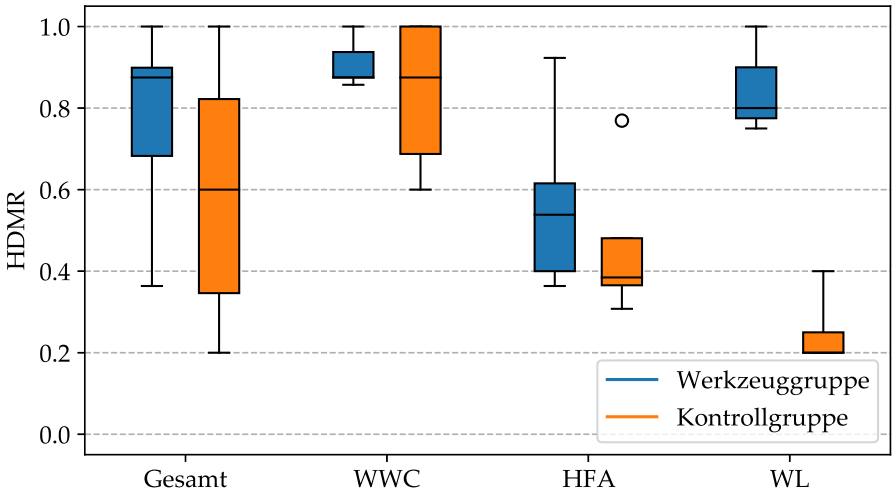


Abbildung 6.12: Experiment 1: Übersehene Fehler ohne Hinweise

dieser Fehler nicht entdeckt. Dennoch werden unter Verwendung des Werkzeugs insgesamt mehr Fehler korrigiert (siehe Abbildung 6.10), da die Fehler ohne Werkzeughinweis nur einen kleinen Teil aller in den Dokumenten vorhandenen Fehler ausmachen (siehe Tabelle 6.6).

Auswertung: Zeit pro Objekt

Die durchschnittliche Zeit, welche die Teilnehmer zur Bearbeitung der Dokumente benötigten, ist in Abbildung 6.13 dargestellt. Es lässt sich beobachten, dass die Zeit pro Objekt durch den Einsatz des Werkzeugs auf dem Scheibenwischersystem sinkt und deutlich konsistenter wird. Ein Großteil der Datenpunkte bewegt sich im Bereich um zehn Sekunden. Am Hands-Free-Access-System lässt sich kein Unterschied beobachten. Die Teilnehmer, die das Werkzeug auf dem Fensterhebersystem einsetzten, wurden hingegen sichtlich langsamer. Dies kann an den vielen False-Positives, bzw. der geringen Genauigkeit des Werkzeugs liegen. Insgesamt verbessert sich die durchschnittliche Zeit pro Objekt etwas, wesentliche Verbesserungen werden jedoch nicht erzielt.

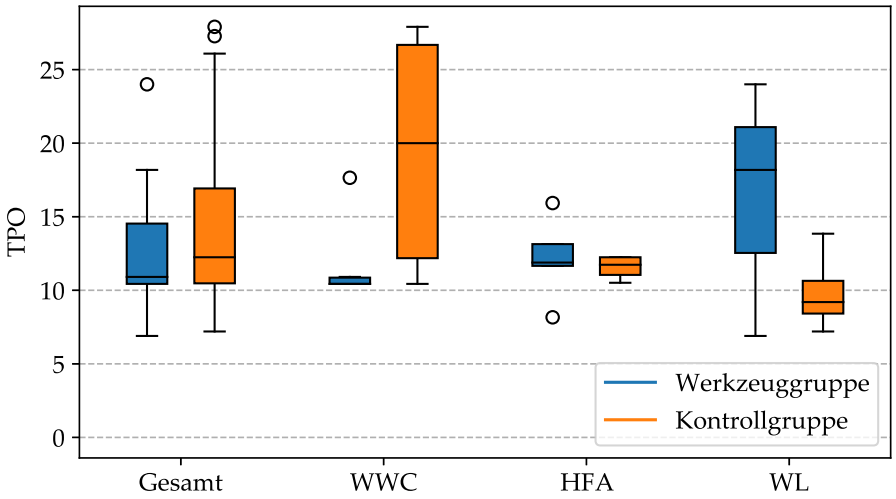


Abbildung 6.13: Experiment 1: Zeit pro Objekt

Obwohl das Werkzeug bestimmte Objekte zur Analyse durch den Benutzer hervorhebt, untersuchten die Benutzer alle anderen Objekte weiterhin auf Fehler.

Auswertung: Umformulierte Objekte

Abbildung 6.14 zeigt zuletzt die Rate umformulierter Objekte. Insgesamt wurden nur von sehr wenigen Teilnehmern Objekte umformuliert. Bei 74% der werkzeuggestützten Reviews ist der Text des gesamten Dokuments unverändert. Bei den manuellen Reviews sind es 70%. Aus den von allen Teilnehmern analysierten Objekten wurden 1,5% umformuliert. Bei 30 der insgesamt 72 geänderten Objekte schlug das Werkzeug fälschlicherweise vor, eine Anforderung als Information zu markieren, da das Objekt wie im folgenden Beispiel zu ungenau formuliert wurde:

> Die Wischzyklen werden von dem Steuergerät selbständig ausgeführt.

Der richtige Objekttyp Anforderung wurde bei diesen 30 Objekten durch den jeweiligen Teilnehmer nicht geändert, stattdessen wurde der Text der Anforderung angepasst:

> Die Wischzyklen müssen von dem Steuergerät selbständig ausgeführt werden.

An dieser Stelle wurde durch das Werkzeug trotz der falschen Hinweise eine Verbesserung der Qualität des Lastenhefts erzielt.

6.2.5 Validität der Ergebnisse

Anzahl der Teilnehmer

Ein Schwachpunkt des durchgeführten Experiments ist die geringe Anzahl an Teilnehmern. Da die Studierenden teilweise in Teams von zwei Personen gearbeitet haben, ist die Anzahl der Ergebnisse nochmals etwas gesunken. Durch die Teams konnten die Studierenden

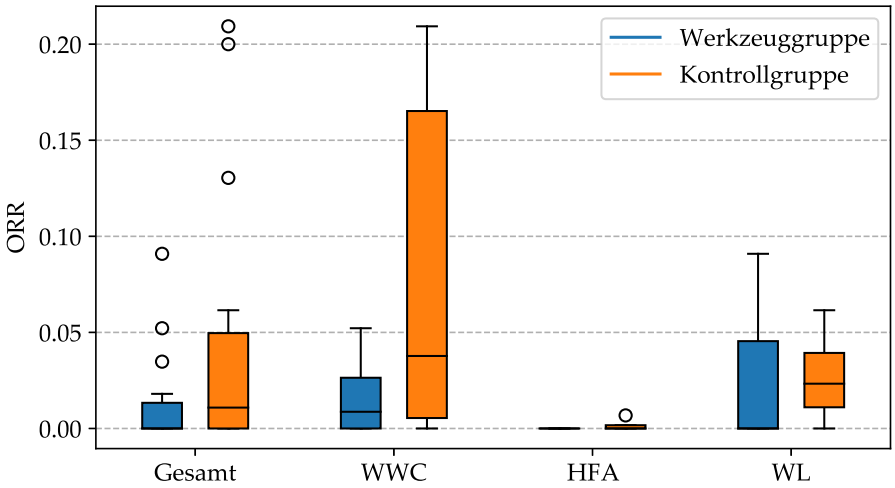


Abbildung 6.14: Experiment 1: Umformulierte Objekte

untereinander Diskussionen führen, was in Qualitätsreviews ebenfalls getan wird. Durch die geringe Teilnehmermenge ist es nicht möglich, statistische Signifikanztests sinnvoll durchzuführen. Eine deutliche Erhöhung der Teilnehmer ist ebenfalls schwierig, da für ein Review sehr viel Zeit benötigt wird und das Experiment zur Sammlung von mehr Datenpunkten aufgrund der Sensibilität der Anforderungsdokumente nicht öffentlich (z.B. online) durchgeführt werden kann.

Definition des Goldstandards

Die Ergebnisse der beiden Review-Methoden wurden gegen einen Goldstandard getestet. Die Definition des Goldstandards hat Auswirkungen auf die Ergebnisse. Die Erstellung des Goldstandards erfolgte von Mitarbeitern des Lehrstuhls Automated System Engineering Technologies (ASET) an der TU Berlin mit mehreren Jahren Erfahrung durch die Kollaboration mit einem Industriepartner. Daher sollte der Goldstandard nahe dem sein, was ein RE-Experte aus der Industrie als korrekt einschätzt.

Unterschiede im Kenntnisstand der Teilnehmer

Das Experiment wurde unter der Annahme durchgeführt, dass die Studierenden abgesehen von den in der Lehrveranstaltung vermittelten Inhalten keine weiterführenden Kenntnisse im Bereich Anforderungsmanagement besitzen. Einige Teilnehmer könnten allerdings mehr Kenntnisse als andere Teilnehmer besitzen und daher bessere Ergebnisse liefern. Dieser Effekt wird dadurch reduziert, dass jeder Teilnehmer beide Review-Methoden angewendet hat.

Lerneffekte

Die Leistung eines Teilnehmers kann sich während des Experiments aufgrund verschiedener Effekte wie Erschöpfung, Sammlung von Erfahrung oder Änderungen in der Motivation verändern. Die Teilnehmer haben möglicherweise im ersten Durchlauf Erfahrungen gesammelt und diese im zweiten Durchlauf angewendet, um so bessere Ergebnisse zu erzielen. Ebenfalls kann ein Verlust an Motivation nach dem ersten Durchlauf negative Auswirkungen auf den zweiten Durchlauf haben.

Gruppenübergreifende Kommunikation

Vor dem Experiment wurde betont, dass zur Wahrung der Unabhängigkeit der einzelnen Ergebnisse eine Kommunikation über die Teams von zwei Personen hinaus nicht erlaubt ist. Da das Experiment in einem großen Raum durchgeführt wurde, kann es sein, dass Informationen zwischen den Gruppen geteilt wurden. Während des Experiments haben sich alle Teilnehmer an diese Auflage gehalten. Dennoch ist nicht sichergestellt, dass alle Ergebnisse tatsächlich unabhängig voneinander sind.

Zeitlimit

Das Zeitlimit wurde aufgrund von zwei Gründen festgelegt: Zum einen ist die Zeit auch in realen Reviews beschränkt und zum anderen musste das Experiment innerhalb von 90 Minuten beendet werden, da es als Teil einer Lehrveranstaltung durchgeführt wurde. Den Teilnehmern wurde mitgeteilt, dass es nicht notwendig ist, das Review innerhalb der Zeit zu beenden. Dennoch können die Studierenden ein vollständiges Review angestrebt haben, dementsprechend weniger Zeit pro Objekt verwendet haben und so ein schlechteres Ergebnis erzielt haben als es ohne Zeitlimit möglich gewesen wäre.

Studierende sind keine RE-Experten

Letztlich sind Studierende im Vergleich zu den Personen, die ein Review tatsächlich durchführen, keine RE-Experten. Insbesondere besitzen sie im Kontrast zu den Autoren der Anforderungsdokumente kein Detailwissen über die Dokumente und kein Wissen über die Prozesse, an denen die Dokumente beteiligt sind. Dies ist jedoch nicht notwendigerweise ein Nachteil: Studierende inspizieren ein Dokument möglicherweise genauer, wohingegen ein RE-Experte mögliche Fehler aufgrund von Prozesseinschränkungen (z.B. die durch Änderungen hervorgerufenen Kosten und zusätzlich benötigte Zeit) ignoriert. In einer Untersuchung von Falessi u. a. [34] wurde gezeigt, dass kontrollierte Experimente mit Studierenden genauso valide sind wie Experimente mit Experten aus der Industrie.

6.2.6 Zusammenfassung

Zusammenfassend werden nun die Forschungsfragen beantwortet.

1. Die Verwendung des Werkzeugs führt zu mehr gefundenen Fehlern, vorausgesetzt die Genauigkeit des Werkzeugs ist hoch und die Anzahl an False-Positives gering. Diese Voraussetzungen waren bei dem Fensterhebersystem nicht gegeben.

2. Wenn das Werkzeug verwendet wird, werden weniger Objekte mit korrekter Objekttyp-Klassifikation irrtümlich geändert.

3. Die Teilnehmer haben bei der Verwendung des Werkzeugs erwartungsgemäß mehr Fehler nicht gefunden, zu denen das Werkzeug keine Warnung ausgegeben hat. Dennoch finden die Teilnehmer mithilfe des Werkzeugs insgesamt mehr Fehler.

4. Die für ein Objekt benötigte Zeit konnte durch das Werkzeug nicht signifikant gesenkt werden. Die durchschnittlich benötigten Zeiten sind durch das Werkzeug jedoch konsistenter und liegen näher am Mittelwert.

5. Insgesamt motivierte das Werkzeug nur wenige Teilnehmer zur Umformulierung der Lastenheftinhalte. Bei den wenigen Änderungen wurde jedoch häufig beobachtet, dass die Änderungen aufgrund falscher Werkzeugvorhersagen bei Objekten mit fraglichen Formulierungen erfolgten.

Grundsätzlich ist das Werkzeug damit in der Lage, hinsichtlich der Klassifikation von Objekten in Anforderungen und Informationen zu qualitativ besseren Anforderungsdokumenten zu führen. Voraussetzung dafür ist eine ausreichend hohe Genauigkeit und eine möglichst geringe Anzahl an False-Negatives des Werkzeugs.

6.3 EXPERIMENT 2: OPTIMIERUNG VON RECALL UND PRECISION

Die Suche nach Fehlern in der Klassifikation von Anforderungen hinsichtlich des Objekttyps ist eine Aufgabe, bei der in einer sehr großen Menge von Objekten wenige Fehler mit hoher Genauigkeit identifiziert werden müssen. Es ist eine Aufgabe, die für einzelne Objekte von Menschen ohne große Probleme durchgeführt werden kann, aber angesichts der großen Menge an Anforderungen in den Spezifikationen nur schwer beherrschbar ist. Vergleichbare Aufgaben sind beispielsweise das Finden von Trace-Links zwischen High-Level-Spezifikationen und Low-Level-Spezifikationen [46, 63] oder zwischen Anforderungs- und Testspezifikationen [86].

Relevanz von Recall in Werkzeugen

In der Literatur werden solche Aufgaben von Daniel M. Berry „hairy requirements or software engineering task“ genannt [20]. Werden solche Aufgaben durch ein Werkzeug unterstützt, so ist es für einen Menschen wesentlich einfacher, einen gefundenen falschen Werkzeughinweis (False-Positive) abzulehnen als einen nicht gefundenen Fehler (False-Negative) dennoch zu finden. Daher ist bei Werkzeugen, die zur Unterstützung dieser Aufgaben eingesetzt werden, Recall in der Regel wichtiger als Precision. Wenn für eine kritische Aufgabe (fast) alle Fehler gefunden werden müssen, so muss das Werkzeug einen Recall von nahe 100% aufweisen können [20].

Zum gemeinsamen Vergleich von Recall und Precision unterschiedlicher Modelle wird oftmals die F_1-Metrik herangezogen. Ist Recall

höher zu gewichten als Precision, kann stattdessen die allgemeine F_β-Metrik verwendet werden:

$$F_\beta = (1 + \beta^2) \times \frac{P \times R}{(\beta^2 \times P) + R} \tag{6.10}$$

Berechnung von β für F_β zur Gewichtung von Recall und Precision

β gibt in dieser Formel an, um wie viel stärker Recall gegenüber Precision bewertet werden soll. Umso höher β, desto weniger relevant ist Precision für die Bewertung eines Modells. β ist ein für jede Aufgabe und damit verbundenes Werkzeug spezifischer Wert und wird aus dem Quotienten der beiden folgenden Werte berechnet [20]:

- Die Zeit, die ein Mensch benötigt, um ohne Werkzeugunterstützung ein True-Positive in einem Dokument zu finden und
- die Zeit, die ein Mensch benötigt, um ein von einem Werkzeug gezeigten False-Positive zu erkennen und zurückzuweisen.

Diese Werte können ausschließlich durch empirische Studien ermittelt werden. In den folgenden Abschnitten wird ein Experiment ähnlich zum dem in Abschnitt 6.2 beschriebenen Experiment vorgestellt, dessen Ergebnisse die Berechnung von β für das Werkzeug und daraufhin weitere Optimierungen und Rückschlüsse auf die Möglichkeiten des Werkzeugs erlauben. Die Ergebnisse des ersten Experiments eignen sich wegen des unpassenden Aufbaus des Experiments nicht für die Berechnung von β. Insbesondere erlaubt das Experiment neben der Bestimmung von Recall und Precision auch die Bestimmung der Summarization des Werkzeugs. Summarization gibt an, um welchen Faktor die Anzahl der zu analysierenden Objekte des Dokuments durch den Werkzeugeinsatz reduziert wird [20]:

Definition von Summarization

$$\mathit{Summarization} = \frac{\text{Anzahl Objekte ohne Warnung}}{\text{Anzahl Objekte}} \tag{6.11}$$

Unter der Annahme, dass das Werkzeug einen genügend hohen Recall auf Fehlern aufweist und dass Objekte ohne Warnungen von Anwendern übersprungen werden, ist höhere Summarization direkt übertragbar auf im Review eingesparte Zeit.

6.3.1 Planung und Durchführung des Experiments

Es wurde erneut ein empirisches Experiment mit Studierenden durchgeführt. Der Aufbau des Experiments musste leicht verändert werden, um die zur Berechnung von β notwendigen Daten zu erheben. In diesem Experiment arbeiten die Teilnehmer in der Werkzeuggruppe nicht direkt mit dem Werkzeug, sondern mit Materialien, die den Materialien der Kontrollgruppe sehr ähnlich sind. Zudem sehen sie

ausschließlich diejenigen Objekte des Dokuments, für die das Werkzeug einen Hinweis gibt.

Wieder wurden die Teilnehmer in zwei Gruppen eingeteilt, von denen jeweils eine mit Werkzeugunterstützung und die andere manuell gearbeitet hat. Das Ziel des Experiments ist die Beantwortung der folgenden drei Forschungsfragen:

Definition der Forschungsfragen für das Experiment

1. *Wie groß ist der Unterschied der Ergebnisse zwischen den beiden Gruppen?* Die Studierenden, die während des Experiments mit dem Werkzeug arbeiten, sollten bessere Ergebnisse erzielen können als die Studierenden, welche die Aufgabe manuell durchführen. Da die Werkzeuggruppe nur einen Ausschnitt des gesamten Dokuments bearbeitet, sollten sie zudem deutlich schneller sein.

2. *Welchen Wert nimmt* β *für Klassifikation nach Anforderungen und Informationen an?* Dieser Wert wird aus Daten berechnet, die durch das Experiment erhoben werden und kann zur Optimierung des Werkzeugs hinsichtlich Recall, Precision und Summarization verwendet werden.

3. *Gegeben* β*, wie groß ist Summarization des Werkzeugs auf Anforderungsdokumenten?* Summarization gibt an, um wie viel die zu analysierende Anzahl an Objekten eines Dokuments durch die Verwendung des Werkzeugs sinkt. Durch eine andere Balance von Recall und Precision wird Summarization beeinflusst, daher werden alle drei Metriken gemeinsam untersucht.

Design des Experiments

Das Experiment wurde wieder als Cross-over-Studie mit zwei Durchläufen und zwei Gruppen gemäß Tabelle 6.5 durchgeführt. Die Teilnehmer wurden in zwei Gruppen unterteilt, wobei im ersten Durchlauf eine Gruppe mit Werkzeugunterstützung und die andere Gruppe ohne Werkzeugunterstützung gearbeitet hat. Im zweiten Durchlauf wurden die Gruppen getauscht.

Teilnehmer

Die Studie wurde wiederum als eine Einheit in der Lehrveranstaltung „Automotive Software Engineering“ an der TU Berlin durchgeführt. Die Teilnehmer besitzen den gleichen Hintergrund und das gleiche Vorwissen wie die Teilnehmer des ersten Experiments. Insgesamt haben 16 Studierende am Experiment teilgenommen. Keiner dieser Studierenden hat am ersten Experiment teilgenommen.

Studienmaterialien

Für das Experiment wurden die beiden Anforderungsdokumente Scheibenwischer (WWC) und Hands-Free-Access (HFA) aus dem vorherigen Experiment wiederverwendet. Für die Durchführung des Experiments wurden diese jedoch anders aufbereitet. In diesem Experiment wurde vollständig auf den Einsatz von Software durch die Teilnehmer verzichtet. Beide Gruppen erhielten die Lastenhefte in Papierform. Dadurch werden Schwankungen in der Zeitmessung eliminiert, die durch Unsicherheit in der Benutzung der jeweiligen

Objekttyp	Objekttext / Überschrift	Korrektur
	2.1.2 Funktionen	
	2.1.2.1 Funktion "»Wischen Frontscheibe«"	
requirement	Die Wischgeschwindigkeit wird vom CBC an die WPRA (UWA) übermittelt	
requirement	Die Wischzyklen werden von dem WPRA (UWA) selbständig ausgeführt.	
	2.1.2.1.1 Reaktionswischen	
information	Bei jedem Hochschalten in eine Intervallstufe wird sofort ein Wischvorgang ausgelöst. Dieser Wischvorgang heißt "Reaktionswischen". Die Definition "Reaktionswischen" ist unabhängig von der Komponente Regensensor.	

Abbildung 6.15: Experiment 2: Ausschnitt des Dokuments der Kontrollgruppe

Objekttyp	Objekttext / Überschrift	Werkzeug-vorschlag	Vorschlag übernehmen?
	2.3 Funktionsbeschreibung		
	2.3.1 HANDS-FREE ACCESS		
	2.3.1.1 Auslöser		
information	Die gesamte Bewegung soll zunächst maximal zwischen 1.0 und 2.0 sek. dauern dürfen, um als erfolgreich detektiert zu werden.	requirement	
information	Für die Funktion von HFA muss der Abstand des auslösenden Objektes für den oberen Sensor (Detektion Schienbein) in x-Richtung 2cm bis 15 cm zur oberen Elektrode (Flach- oder Rundleiter) betragen.	requirement	

Abbildung 6.16: Experiment 2: Ausschnitt des Dokuments der Werkzeuggruppe

Werkzeuge (Microsoft Excel bzw. das in Kapitel 6.2.1 vorgestellte Werkzeug) verursacht werden.

Die Kontrollgruppe erhielt das vollständige Lastenheft. Für jedes Objekt des Lastenhefts wurde ein für die Auswertung benötigter Schlüssel, der Anforderungstext, der im Dokument vorhandene Objekttyp und ein Freifeld für Korrekturen bereitgestellt. Abbildung 6.15 zeigt einen Ausschnitt des Dokuments.

Die Werkzeuggruppe erhielt ein Dokument, in dem lediglich diejenigen Objekte des originalen Lastenhefts enthalten sind, für die das Werkzeug eine Änderung vorgeschlagen hat. Das Dokument enthält für jedes Objekt den vom Werkzeug vorgeschlagenen Objekttyp und ein Freifeld für die Korrekturen der Teilnehmer. Außerdem enthält das Dokument zur besseren Verständlichkeit alle relevanten Überschriften. Überschriften, unter denen sich keine Objekte mit Werkzeuganmerkungen befinden, wurden ausgeblendet. Abbildung 6.16 zeigt einen Ausschnitt des Dokuments.

Tabelle 6.7 zeigt eine Zusammenfassung der Dokumente, darunter auch Summarization, Recall und Precision des Werkzeugs auf beiden Dokumenten. Die Dokumente unterscheiden sich zu den Dokumenten des ersten Experiments in der Anzahl der Objekte und Fehler, da minimale Änderungen an den Dokumenten und dem Goldstandard vorgenommen wurden. Die Anzahl und Korrektheit der Werkzeughinweise unterscheidet sich ebenfalls, da für dieses Experiment ein neues Klassifikationsmodell verwendet wurde.

Aufgaben

Die Aufgaben für das Experiment orientieren sich an den Aufgaben, die RE-Experten in einem Review durchführen. Teilnehmer der Kon-

METRIK	WWC	HFA
Anzahl Objekte	115	148
Anzahl Anforderungen	85	80
Anzahl Informationen	30	68
Anzahl Fehler	19	47
Fehler je Objekt	0,165	0,320
Anzahl Werkzeughinweise	25	37
Richtige Werkzeughinweise	16	31
Summarization	$1 - \frac{25}{115} = 78,3\%$	$1 - \frac{37}{147} = 74,8\%$
Recall	$\frac{16}{19} = 0,842$	$\frac{31}{47} = 0,660$
Precision	$\frac{16}{25} = 0,640$	$\frac{31}{37} = 0,838$

Tabelle 6.7: Experiment 2: Zusammenfassung der Dokumente

trollgruppe sollten das Anforderungsdokument lesen und für jedes Objekt entscheiden, ob der angegebene Objekttyp richtig ist. Bei Bedarf sollen die Teilnehmer Änderungen im Korrekturfeld vornehmen. Die Teilnehmer der Werkzeuggruppe entscheiden für jedes Objekt des Anforderungsdokuments, ob die vom Werkzeug vorgeschlagene Änderung übernommen werden soll. Dies soll im Korrekturfeld durch ein X bestätigt werden.

Ablauf des Experiments

Wie auch das erste Experiment wurde dieses Experiment als Teil einer Lehrveranstaltung durchgeführt. Für das Experiment standen 90 Minuten Zeit zur Verfügung.

EINFÜHRUNG (25 MINUTEN). Zu Beginn des Experiments wurde den Teilnehmern das Problem der Klassifikation von Lastenheftinhalten in Anforderungen und Informationen erklärt. Zu beiden Klassen wurden Beispiele gegeben und die Auswirkungen falscher Klassifikation auf die dem Anforderungsmanagement folgenden Entwicklungsprozesse dargestellt. Weiterhin wurde das Experiment, die Ziele des Experiments und die Aufgabe erklärt, welche die Teilnehmer durchführen sollten.

DURCHLAUF 1 (20 MINUTEN). Die Anforderungsdokumente wurden bereits im Voraus auf den Tischen verteilt und teilten die Studierenden in zwei gleich große Gruppen. Im ersten Durchlauf arbeiteten die Studierenden am Dokument des Scheibenwischersystems. Die Teilnehmer durften nicht untereinander kommunizieren, da sonst die Unabhängigkeit der Ergebnisse beeinträchtigt wird.

DURCHLAUF 2 (25 MINUTEN). Im zweiten Durchlauf arbeiteten die Teilnehmer am Hands-Free-Access-Dokument. Die Gruppen-

	KONTROLLGRUPPE	WERKZEUGGRUPPE
Anzahl Reviews	16	16
Objekte untersucht	2.096	496
Objekte geändert	448	354
Fehler korrigiert	216	274
Summe Gesamtzeit	18.609 s	6.913 s

Tabelle 6.8: Experiment 2: Überblick über die Ergebnisse

zugehörigkeit jedes Teilnehmers wurde für diesen Durchlauf getauscht.

AUSBLICK (15 MINUTEN). Nach beiden Durchläufen wurden die Ergebnisse eingesammelt und es wurde gezeigt, wie die Ergebnisse ausgewertet werden und die Forschung am Einsatz von maschinellem Lernen in Werkzeugen beeinflussen können.

Pilotierung

Auch bei diesem Experiment wurde ein Pilottest durchgeführt. In diesem Test wurde das Experiment mit Mitarbeitern des Lehrstuhls Automated System Engineering Technologies (ASET) an der TU Berlin durchgeführt. Anhand der Ergebnisse wurde verifiziert, dass der Aufbau des Experiments dazu geeignet ist, β zu berechnen und relevante Auswertungen auf den Ergebnissen durchführen zu können.

Datenverfügbarkeit

Die Daten, die während dieses Experiments erhoben wurden, sind wie auch die Daten des ersten und zweiten Experiments online verfügbar [112].

6.3.2 Metriken zur Auswertung der Ergebnisse

Die Auswertung der Ergebnisse erfolgt in drei Schritten. Zuerst werden die Ergebnisse der Kontrollgruppe und der Werkzeuggruppe verglichen und überprüft, inwiefern die Benutzung des Werkzeugs zu besseren Review-Ergebnissen führt. Daraufhin wird β auf Basis der Ergebnisse berechnet. Als letztes wird mithilfe von F_β Precision, Recall und Summarization des Klassifikators optimiert.

Überblick über die Ergebnisse

Tabelle 6.8 zeigt einen Überblick über die Ergebnisse. Insgesamt wurden 32 Reviews durchgeführt, davon eine Hälfte durch die Kontrollgruppe und die andere Hälfte durch die Werkzeuggruppe. In diesen Daten ist bereits eine Tendenz erkennbar: die Werkzeuggruppe hat deutlich weniger Objekte untersucht, mehr Fehler korrigiert und verwendete dafür deutlich weniger Zeit. Die beiden Gruppen werden anhand der folgenden Metriken verglichen:

Definition der Metriken für die Auswertung

Der *Recall* gibt an, wie viele der im Dokument vorhandenen Fehler durch die Teilnehmer korrigiert wurden.

$$\text{Recall} = \frac{\text{Anzahl Fehler korrigiert}}{\text{Gesamtanzahl Fehler}} \tag{6.12}$$

Der Recall der Werkzeuggruppe kann niemals höher sein als der Recall des Werkzeugs (vgl. Tabelle 6.7), da die Teilnehmer dieser Gruppe nur die durch das Werkzeug erkannten Fehler finden können. Der Recall wird auf Basis aller Fehler ermittelt, die im Goldstandard definiert sind. Teilnehmer der Kontrollgruppe haben die Möglichkeit, alle Fehler zu finden. Durch den Fokus auf wenige und wahrscheinlich fehlerhafte Objekte sollte die Werkzeuggruppe mehr Fehler korrigieren können als die Kontrollgruppe:

$$H_1 : \text{Recall}(\text{WG}) > \text{Recall}(\text{KG}) \tag{6.13}$$

Precision gibt an, wie viele der von den Teilnehmern geänderten Objekte einen per Goldstandard definierten Fehler korrigieren.

$$\text{Precision} = \frac{\text{Anzahl Fehler korrigiert}}{\text{Objekte geändert}} \tag{6.14}$$

Ein Objekt gilt als geändert, wenn der Teilnehmer dem Objekt einen anderen Objekttyp zugewiesen hat oder die durch das Werkzeug vorgeschlagene Änderung akzeptiert hat. Wird ein korrekt markiertes Objekt geändert, erzeugt der Teilnehmer einen neuen Fehler. Ist die Precision geringer als 0,5, werden durch den Teilnehmer mehr Fehler erzeugt als korrigiert. Auch die Precision der Werkzeuggruppe sollte durch den Fokus auf wenig und sehr wahrscheinlich fehlerbehaftete Objekte besser sein als die Precision der Kontrollgruppe:

$$H_2 : \text{Precision}(\text{WG}) > \text{Precision}(\text{KG}) \tag{6.15}$$

Die *Zeit pro Objekt* gibt an, wie lange ein Teilnehmer durchschnittlich verwendet hat, um eine Entscheidung für ein Objekt zu treffen. Diese Zeit umfasst das Lesen und Verstehen des Objekts, die Entscheidungsfällung und das Dokumentieren der Entscheidung in der Spezifikation.

$$\text{Zeit pro Objekt} = \frac{\text{Gesamtzeit}}{\text{Gesamtanzahl Objekte}} \tag{6.16}$$

Die Teilnehmer der Werkzeuggruppe werden wahrscheinlich mehr Zeit pro Objekt benötigen, da sie im Entscheidungsprozess ebenfalls den Werkzeugvorschlag berücksichtigen müssen:

$$H_3 : \text{Zeit pro Objekt}(\text{WG}) > \text{Zeit pro Objekt}(\text{KG}) \tag{6.17}$$

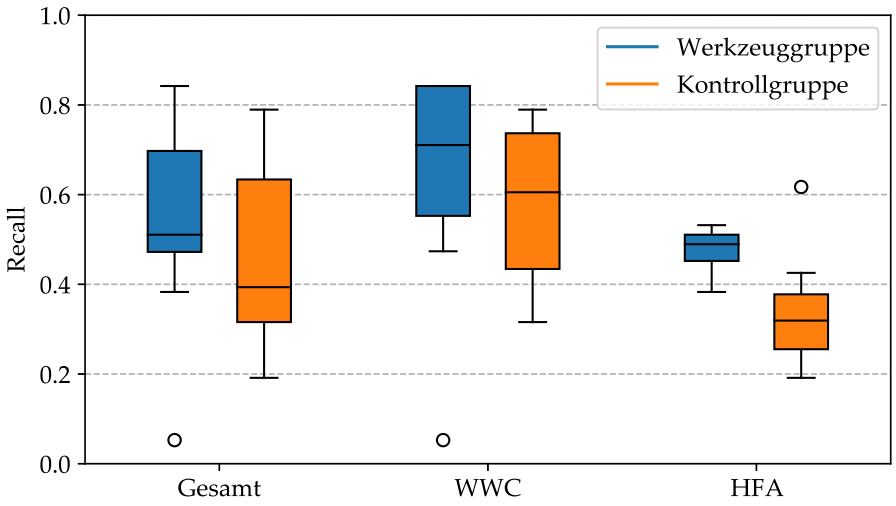


Abbildung 6.17: Experiment 2: Recall

Die *Gesamtzeit* misst, wie lange ein Teilnehmer für die Bearbeitung der Spezifikation benötigt hat. Diese Zeit sollte bei der Werkzeuggruppe deutlich geringer sein:

$$H_4 : \text{Gesamtzeit}(WG) < \text{Gesamtzeit}(KG) \tag{6.18}$$

6.3.3 Auswertung der Ergebnisse

Auswertung: Recall

Abbildung 6.17 zeigt den Recall beider Gruppen. Auf beiden Dokumenten ist der Recall der Werkzeuggruppe höher. Im Mittel steigt der Recall von 0,39 auf 0,51. Obwohl das Werkzeug nicht alle Fehler findet, resultierte dessen Verwendung dennoch in einer höheren Anzahl gefundener Fehler. Dadurch wird Hypothese H_1 bestätigt. Die Ergebnisse auf der Hands-Free-Access-Spezifikation sind deutlich schlechter als die Ergebnisse der Scheibenwischer-Spezifikation. Dies liegt möglicherweise an schwieriger zu verstehenden Anforderungen oder an der deutlich größeren Anzahl an Fehlern. Ein einziges Review der Werkzeuggruppe von der Scheibenwischer-Spezifikation weist einen besonders niedrigen Recall auf: Dieser Teilnehmer hat in den 25 gegebenen Werkzeughinweisen fünf Änderungen durchgeführt und mit diesen Änderungen lediglich einen von 19 Fehlern korrigiert.

Insgesamt übertraf kein Teilnehmer in der Kontrollgruppe den durch das Werkzeug limitierten größtmöglichen Recall der Werkzeuggruppe. Dies zeigt, dass sich ein Recall von weniger als 100% nicht negativ auf die Ergebnisse der Reviews auswirkt.

Auswertung: Precision

Die Precision (siehe Abbildung 6.18) stieg durch die Verwendung des Werkzeugs deutlich an. Teilnehmer der Kontrollgruppe erreichten durchschnittlich eine Präzision von 0,5. Nur etwa die Hälfte der Änderungen der Teilnehmer korrigierte einen Fehler, die andere Hälfte

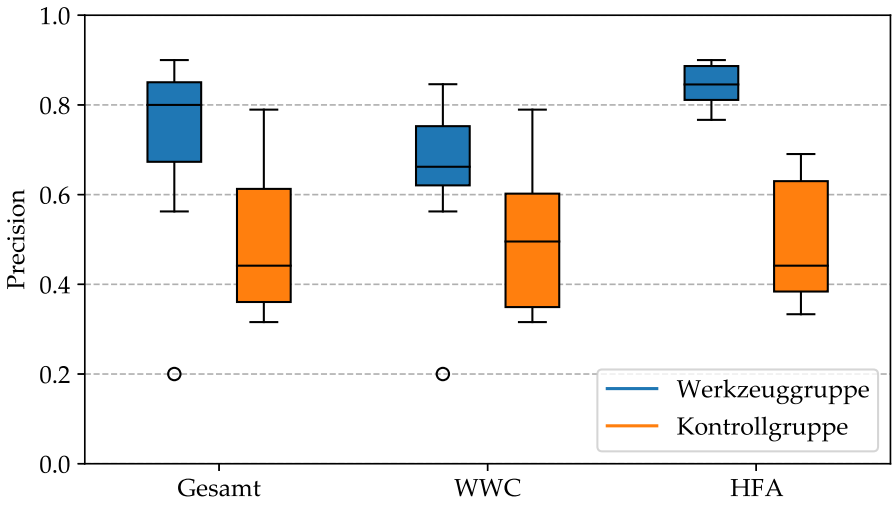


Abbildung 6.18: Experiment 2: Precision

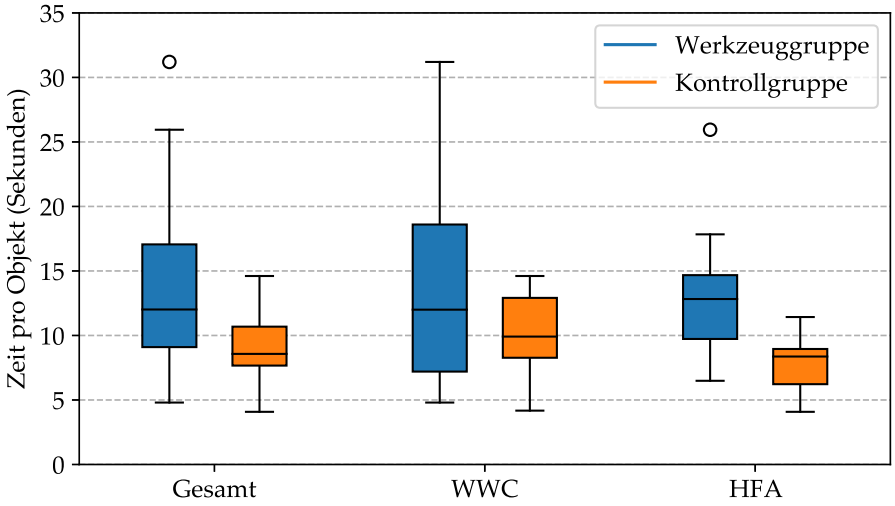


Abbildung 6.19: Experiment 2: Zeit pro Objekt

der Änderungen führte zu neuen Fehlern in den Dokumenten. Der Kontrollgruppe gelang es damit nicht, die Qualität der Dokumente zu verbessern. Die Ergebnisse der Werkzeuggruppe sind deutlich besser, hier liegt die durchschnittliche Precision bei 0,8. Dadurch bestätigt sich Hypothese H_2.

Auswertung: Zeit pro Objekt

Die Zeit pro Objekt ist in Abbildung 6.19 dargestellt und liegt bei der Kontrollgruppe bei etwa neun Sekunden und bei der Werkzeuggruppe bei etwa zwölf Sekunden. Dies sind ähnliche Zeiten, die auch bei dem in Abschnitt 6.2 beschriebenen Experiment ermittelt wurden. Durchschnittlich verwenden Teilnehmer der Werkzeuggruppe mehr Zeit für ein Objekt als Teilnehmer der Kontrollgruppe, wodurch Hypothese H_3 bestätigt wird. Ein möglicher Grund dafür sind die Werkzeughinweise, die die Studierenden ebenfalls auswerten müssen. Die Teilnehmer sind sich zudem bewusst, dass jedes vom Werkzeug zurückgegebe-

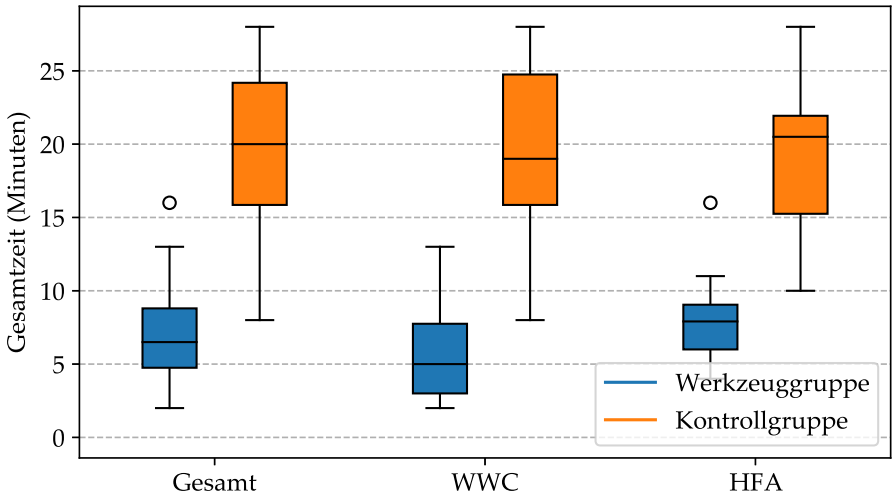


Abbildung 6.20: Experiment 2: Gesamtzeit

ne Objekt sehr wahrscheinlich fehlerhaft ist und untersuchen diese Objekte daher genauer. Teilnehmer der Kontrollgruppe können viele offensichtlich richtig klassifizierte Objekte schneller überfliegen.

Auswertung: Gesamtzeit

Abbildung 6.20 zeigt die Gesamtzeit, welche die Teilnehmer für das Review benötigt haben. Die Gesamtzeit beider Gruppen der Hands-Free-Access-Spezifikation ist etwas größer, da diese Spezifikation umfangreicher ist als die Scheibenwischer-Spezifikation. Die Grafik zeigt, dass die Teilnehmer der Werkzeuggruppe trotz der höheren Zeit pro Objekt das Review des Dokuments in etwa einem Drittel der Zeit der Kontrollgruppe abschließen konnten. Dies bestätigt Hypothese H_4.

Insgesamt zeigen die Ergebnisse, dass die Verwendung des Werkzeugs zur Unterstützung des Reviews zu besseren Ergebnissen bei der Erkennung von Fehlern führt und gleichzeitig die für das Review benötigte Zeit signifikant senkt.

6.3.4 Berechnung von β

Mithilfe der mit dem Experiment ermittelten Ergebnisse kann nun β wie von Berry [20] beschrieben ermittelt werden:

$$\beta = \frac{\text{Gesamtzeit}_{KG}}{\#\text{ Fehler korrigiert}_{KG}} \times \frac{1}{\text{Zeit pro Objekt}_{WG}} \tag{6.19}$$

$$= \frac{\sum_{k\ in KG} \text{Zeit}_k}{\#\text{ Fehler korrigiert}_{KG}} \times \frac{\#\text{ Objekte untersucht}_{WG}}{\sum_{w\ in WG} \text{Zeit}_w} \tag{6.20}$$

$$= \frac{18\,609\,\text{s}}{216} * \frac{496}{6913\,\text{s}} \tag{6.21}$$

$$= 6,18 \tag{6.22}$$

$$\approx 6,2 \tag{6.23}$$

In dieser Berechnung ist der erste Part der Formel 6.19 die durchschnittliche Zeit, die ein Teilnehmer der Kontrollgruppe (KG) benötigt, um ohne Werkzeugunterstützung einen Fehler zu identifizieren und zu korrigieren. Der zweite Part ist die durchschnittliche Zeit, die ein Teilnehmer der Werkzeuggruppe (WG) benötigt, um einen Werkzeughinweis zu untersuchen und entweder zu akzeptieren oder zurückzuweisen.

Abschätzen der Genauigkeit von β

Die Berechnung von β basiert auf den während des Experiments von den Teilnehmern durchgeführten Zeitmessungen. Die Genauigkeit kann daher nur so gut sein wie die Genauigkeit der zugrundeliegenden Messungen. Da viele Teilnehmer die Zeitmessung lediglich in Minuten vorgenommen haben, wird für alle Zeitmessungen ein Fehler von $\pm 30\,s$ angenommen. Mithilfe von Fehlerpropagation kann damit der Fehler von β wie folgt berechnet werden:

$$\delta\beta = \sqrt{\left(\frac{\sqrt{(\pm 30\,s)^2 \times 16}}{18\,609\,s}\right)^2 + \left(\frac{\sqrt{(\pm 30\,s)^2 \times 16}}{6913\,s}\right)^2} \tag{6.24}$$

$$= \pm 0,0185 \tag{6.25}$$

$$\Delta\beta = \pm 0,0185 \times 6,18 \tag{6.26}$$

$$= \pm 0,11 \tag{6.27}$$

Die beiden Zeiten in Formel 6.21 sind Summen aller individuellen Zeitmessungen. Der Fehler dieser beider Werte ist jeweils die Wurzel der Summe aller quadrierten absoluten Fehler. Der relative Fehler $\delta\beta$ wird aus den relativen Fehlern dieser beiden Zeiten berechnet. Der absolute Fehler $\Delta\beta$ ist demnach 0,11.

Für die Suche nach Fehlern in der Objekttypklassifikation liegt β demnach im Bereich von etwa 6,0 bis 6,3. Die Leistung des Klassifikators kann mit $F_{6,2}$ besser bewertet werden als mit F_1. Dies bedeutet insbesondere, dass Recall für die Bewertung der Leistung deutlich wichtiger ist als Precision.

6.3.5 Optimierung von Precision, Recall und Summarization

Ausbalancieren von Recall und Precision durch einen Schwellenwert

Mithilfe des berechneten β wird nun der Klassifikator optimiert. Der Klassifikator kann mithilfe eines Schwellenwerts so konfiguriert werden, dass entweder wenige und wahrscheinlich richtige Warnungen oder viele und möglicherweise falsche Warnungen ausgegeben werden. Je nach gewähltem Schwellenwert ist damit bei hohem Schwellenwert Recall gering und Precision hoch, oder bei niedrigem Schwellenwert Recall hoch und Precision gering. Beide Extremfälle, in denen jeweils alle Objekte oder kein einziges Objekt als fehlerhaft ausgegeben werden, sind ebenfalls möglich. Da F_β auf Basis von Recall und Precision berechnet wird, wirkt sich die Wahl des Schwellenwerts

ebenfalls auf F_β aus. Es wird derjenige Schwellenwert gesucht, für den F_β maximal ist.

Die Wahl des Schwellenwerts wirkt sich ebenfalls auf Summarization des Klassifikators aus. Umso höher der Schwellenwert gewählt wird, desto weniger Werkzeughinweise werden gegeben und desto größer ist demnach Summarization. Durch die Optimierung hinsichtlich F_β wird daher ebenfalls ermittelt, wie hoch die durchschnittlich zu erwartende Zeitersparnis durch den Einsatz des Werkzeugs ist.

Precision und Recall für Fehlererkennung ermitteln

Der Klassifikator klassifiziert Objekte in Anforderungen und Informationen. Der Datensatz enthält entsprechend klassifizierte Objekte. Durch die Zerlegung des Datensatzes in einen Trainingsdatensatz und einen Testdatensatz ist es möglich, Precision und Recall für diese beiden Klassen zu ermitteln. Um Precision, Recall und damit ebenfalls F_β für die Erkennung von Fehlern zu ermitteln, müssen im Datensatz Objekte als fehlerbehaftet definiert und darauffolgend die Fähigkeit des Klassifikators gemessen werden, diese und genau nur diese Objekte zu identifizieren.

Dazu wird vorerst folgende Annahme getroffen: Die im Datensatz enthaltenen Objekte sind korrekt klassifiziert. Trotz der während der Erstellung des Datensatzes vorgenommenen Maßnahmen zur Entfernung falsch klassifizierter Objekte werden dennoch einige solche Objekte im Datensatz enthalten sein. Wird die Leistung des Klassifikators am Testdatensatz gemessen, muss der Klassifikator den Objekttyp richtig klassifizierter Objekte richtig bestimmen. Wurde bei einem Objekt im Testdatensatz der Objekttyp nachträglich geändert (d.h., das Objekt ist fehlerbehaftet), so muss der Klassifikator diese Abweichung erkennen.

Der Klassifikator wurde erneut mittels Kreuzvalidierung evaluiert. Dabei wurden in jedem der zehn Durchläufe bei zufällig gewählten Objekten im Testdatensatz Fehler erzeugt und die Fähigkeit des Klassifikators gemessen, diese Fehler zu erkennen. Dadurch lassen sich Precision und Recall der Erkennung von Fehlern messen. Die Fehlerrate wurde entsprechend der im Experiment verwendeten Dokumente auf 25% gesetzt (siehe Tabelle 6.7, Durchschnitt beider Dokumente).

Auswertung: Precision, Recall und Summarization

Abbildung 6.21 zeigt Precision, Recall und Summarization des Klassifikators. Wie erwartet ist es nicht möglich, sowohl hohen Recall als auch hohe Precision zu erreichen. Wenn der Schwellenwert so gewählt wird, dass Recall bei etwa 0,95 liegt, sinkt Precision auf etwa 0,5. Bei hoher Precision sinkt Recall signifikant.

Summarization ist ebenfalls vom gewählten Schwellenwert abhängig. Umso höher die Ansprüche an Recall sind, desto geringer ist die durch das Werkzeug erreichte Summarization und die damit bei Reviews eingesparte Zeit. Der Graph zeigt allerdings auch, dass bei einem Schwellenwert von etwa 0,4 ein RE-Experte nur noch die Hälfte aller Objekte in einem Dokument analysieren muss und in diesen Objekten 98% aller Fehler finden wird.

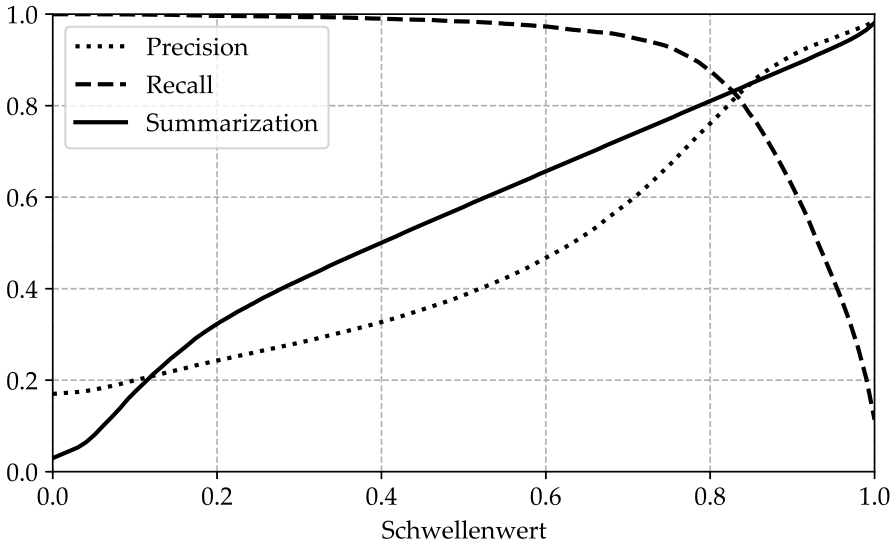


Abbildung 6.21: Experiment 2: Precision, Recall und Summarization in Abhängigkeit eines Schwellenwerts

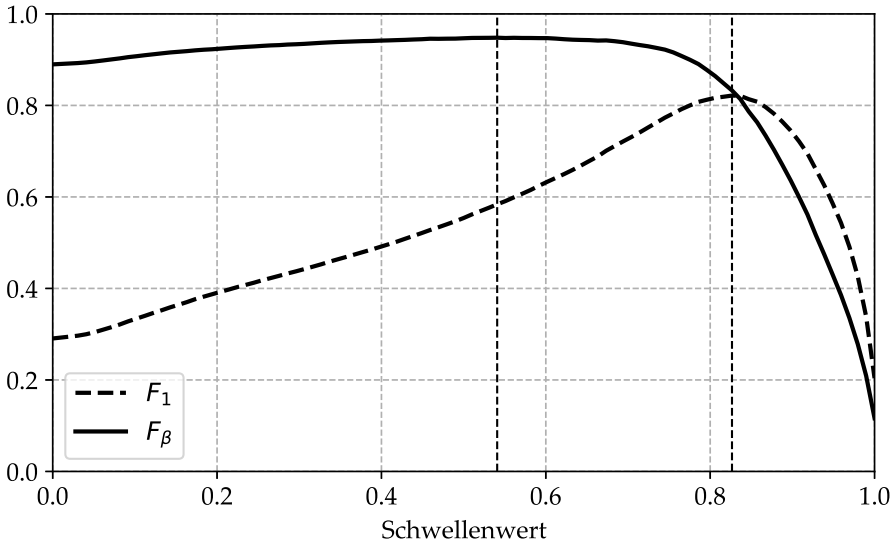


Abbildung 6.22: Experiment 2: F_1 und F_b in Abhängigkeit eines Schwellenwerts

Abbildung 6.22 zeigt F_1 und F_β in Abhängigkeit vom Schwellenwert. Die vertikalen Markierungen zeigen jeweils das Maximum von F_1 und F_β. Das Maximum von F_1 wird bei einem Schwellenwert von 0,83[1] erreicht und führt zu einem Recall von 0,83, einer Precision von 0,81 und einer Summarization von 0,83. Wird stattdessen F_β verwendet, erreicht der Klassifikator bei einem Schwellenwert von 0,54 einen Recall von 0,98, eine Precision von 0,42 und eine Summarization von 0,61. Für die Aufgabe, in Anforderungsdokumenten Fehler in der Objekttypklassifikation zu finden, stellt dieser Kompromiss aus Recall und Precision das Optimum dar. Eine weitere Erhöhung des Recalls würde die Vorteile hoher Summarization zunichtemachen, während eine Erhöhung von Precision und Summarization die Nützlichkeit des Werkzeugs durch zu viele nicht erkannte Fehler beeinträchtigt.

6.3.6 Validität der Ergebnisse

Die Validität der Ergebnisse des Experiments wird eventuell durch die folgenden Punkte eingeschränkt.

Teilnehmer

Studierende sind keine RE-Experten. Sie haben in der Regel weniger Kenntnisse über die Anforderungsmanagement-Prozesse als RE-Experten und weniger Kenntnisse über die im Experiment verwendeten Spezifikationen als deren jeweilige Autoren. Daher können empirische Studien mit RE-Experten statt Studierenden zu anderen Ergebnissen führen.

- RE-Experten sind wahrscheinlich schneller in der Durchführung des Reviews. Sie werden sowohl mit als auch ohne das Werkzeug weniger Zeit benötigen. Gemäß Formel 6.20 wird dies keine signifikanten (d.h., $\beta \approx 1$ oder $\beta > 10$) Auswirkungen auf die Berechnung von β haben.
- RE-Experten werden wahrscheinlich ohne Werkzeugunterstützung mehr Fehler identifizieren können als Studierende. Gemäß Formel 6.20 führt dies aufgrund der höheren Anzahl korrigierter Fehler in der Kontrollgruppe zu einem kleineren β.

Anzahl der Ergebnisse

Weiterhin wurde die statistische Signifikanz der Ergebnisse nicht untersucht, da hierfür zu wenig Ergebnisse vorliegen. Obwohl zwischen beiden Spezifikationen mit jeweils getauschten Gruppen Muster erkennbar sind, können die Ergebnisse einer Wiederholung des Experiments mit anderen Teilnehmern deutlich anders ausfallen. Es ist schwer, Evaluationen mit erheblich größeren Gruppen durchzuführen, da die für das Experiment verwendeten Spezifikationen trotz Anonymisierung nicht veröffentlicht werden können und demnach Online-Evaluationen nicht möglich sind.

1 Schwellenwerte sind jeweils nur für einen trainierten Klassifikator gültig und sind von diversen Hyperparametern wie Epochenanzahl und Netzwerkkonfiguration abhängig.

Lerneffekte

Es ist möglich, dass die Teilnehmer während des Experiments ihr Verhalten aufgrund von während des Experiments gesammelten Erfahrungen, Erschöpfung oder Änderungen in der Motivation verändern. In diesem Experiment können die Teilnehmer die im ersten Durchlauf gesammelten Erfahrungen für das Review im zweiten Durchlauf verwendet haben und somit andere Ergebnisse erzielt haben.

Zeitbegrenzung

Alle Studierenden haben das Review innerhalb der vorgegebenen Zeit abgeschlossen. Einige Teilnehmer waren sogar deutlich schneller als erwartet fertig. Das Zeitlimit kann die Teilnehmer unter Druck gesetzt haben und somit zu schlechteren Ergebnissen geführt haben. Dies gilt insbesondere für die Kontrollgruppe, da diese Gruppe wesentlich mehr Objekte untersuchen musste.

Goldstandard

Der zur Auswertung des Experiments verwendete Goldstandard wurde am Lehrstuhl Automated System Engineering Technologies (ASET) an der TU Berlin erstellt und nicht durch RE-Experten. Der Goldstandard spiegelt daher möglicherweise nicht die absolute Wahrheit wider.

6.3.7 Zusammenfassung

Wenn RE-Experten ein Review einer Spezifikation durchführen, untersuchen sie jedes Objekt der Spezifikation und überprüfen es anhand diverser Richtlinien und Kriterien. Ein Großteil der Objekte erfüllt jedoch diese Kriterien bereits und bedarf daher keiner weiteren Untersuchung. Die RE-Experten verwenden allerdings dennoch Zeit, um diese Objekte zu untersuchen. Durch den Einsatz des Werkzeugs kann diese Zeit drastisch reduziert werden, indem viele höchstwahrscheinlich regelkonforme Objekte ausgeblendet werden.

Insgesamt liefert das Experiment gute Ergebnisse. Diese werden nun anhand der anfangs gestellten Forschungsfragen zusammengefasst.

1. Auf beiden Dokumenten ist die Leistung der Werkzeuggruppe besser als die Leistung der Kontrollgruppe. Teilnehmer der Werkzeuggruppe haben 31% mehr Fehler gefunden und benötigten dafür 63% weniger Zeit. Auch die Precision der Werkzeuggruppe ist deutlich besser.

2. Auf Basis der Ergebnisse des Experiments ist $\beta = 6,2$ optimal.

3. Bei der Objekttypklassifikation ist das Werkzeug in der Lage, Die Anzahl der zu inspizierenden Objekte um durchschnittlich 61% zu senken. Es stellt dabei sicher, dass in den verbleibenden Objekten 98% aller Fehler enthalten sind. Dies variiert natürlich je nach Spezifikation. Summarization wird auf Spezifikationen mit fraglicher Qualität geringer sein.

7 ANWENDUNGSFALL: PVM-KLASSIFIKATION

Verifizierbarkeit ist eine von vielen Qualitätseigenschaften von Anforderungen [52]. Eine Anforderung ist verifizierbar, wenn die Anforderung es ermöglicht, die Erfüllung der in der Anforderung spezifizierten Eigenschaft im System nachzuweisen. Anforderungen sollten verifizierbar sein, da die Spezifikation von Anforderungen, deren korrekte Umsetzung in einem System nicht nachgewiesen werden kann, sinnlos ist.

Unterschiedliche Methoden zur Verifikation von Anforderungen

Anforderungen können auf unterschiedliche Art und Weise verifiziert werden. Mögliche Methoden umfassen unter anderem manuelle Überprüfungen, Prozessaudits, formale Beweise oder automatisierte Tests (z.B. HiL). Jede dieser Methoden ist für bestimmte Arten von Anforderungen geeignet. Die Wahl einer Verifikationsmethode beeinflusst ebenfalls, wie Anforderungen in sukzessiven Prozessschritten behandelt werden [32]. Daher wird für jede Anforderung festgelegt, welche Verifikationsmethode für diese Anforderung anzuwenden ist.

In den in dieser Arbeit untersuchten Anforderungsdokumenten existiert für diesen Zweck ein separates Attribut *Empfohlene Verifikationsmethode* (engl. „Potential Verification Method“, PVM). Für jede Anforderung spezifiziert dieses Attribut, welche Verifikationsmethode für eine Anforderung anzuwenden ist. Genauso wie das Attribut Objekttyp wird auch das Attribut PVM manuell gesetzt und in Reviews manuell überprüft. Daher liegt es nahe, das Werkzeug zur Objekttypklassifikation ebenfalls zur Unterstützung der Klassifikation des PVM-Attributs einzusetzen.

In den Kapiteln 4, 5 und 6 wurde ausführlich gezeigt, wie neuronale Netzwerke zur Unterstützung von Entwicklern bei der Klassifikation von Lastenheftinhalten in Anforderungen und Informationen eingesetzt werden können. In diesem Kapitel wird ein neuronales Netzwerk zur Klassifikation von Anforderungen in empfohlene Verifikationsmethoden trainiert. Durch die Übertragung auf das Attribut PVM wird gezeigt, dass der Ansatz nicht auf die Klassifikation von Informationen und Anforderungen beschränkt ist und auch auf andere Anwendungsfälle übertragbar ist.

7.1 EMPFOHLENE VERIFIKATIONSMETHODE

Das Attribut Empfohlene Verifikationsmethode einer Anforderung ist eine Empfehlung für die Bearbeitung in nachfolgenden Prozessschrit-

ten, welche Verifikationsmethode für eine Anforderung angewendet werden soll. Die Verifikationsmethoden können je nach Domäne unterschiedlich sein, in den in dieser Arbeit untersuchten Dokumenten aus der Automobilindustrie wurden die folgenden sechs Verifikationsmethoden verwendet.

Übersicht empfohlene Verifikationsmethoden

SYSTEMTEST. Eine Anforderung, deren Erfüllung durch einen manuellen oder automatisierten Test am implementierten System überprüft wird. Dies können z.B. Hardware-in-the-Loop-Tests sein. Diese Verifikationsmethode eignet sich für alle Anforderungen, die Systemeigenschaften beschreiben und wird für den Großteil der Anforderungen eines Lastenhefts verwendet.

REVIEW. Manuelle Überprüfung der Erfüllung einer Anforderung durch Review eines Entwicklungsartefakts, beispielsweise Code und Architekturentwürfe. Anforderungen, die Eigenschaften eines entwicklungsbegleitenden Artefakts spezifizieren, können oftmals nicht am implementierten System überprüft werden. Für diese Anforderungen eignet sich die Verifikationsmethode Review.

SIMULATION/ANALYSIS. Insbesondere bei Anforderungen an die Software von Steuergeräten können die in einer Anforderung spezifizierten Systemeigenschaften auch durch eine Simulation der Steuergerätesoftware und dessen Umgebung verifiziert werden. Dafür existiert die Verifikationsmethode Simulation/Analysis.

FORMAL VERIFICATION. In einer formalen Überprüfung von Systemeigenschaften wird die Erfüllung einer Anforderung durch Anwendung mathematischer Methoden (z.B. Beweise) sichergestellt.

PROCESS AUDIT. Anforderungsdokumente stellen nicht nur Anforderungen an das System, sondern auch an die eingesetzten Entwicklungsprozesse (vgl. Abschnitt 2.1, Rahmenbedingungen). Die Erfüllung dieser Anforderungen kann durch ein Prozessaudit sichergestellt werden.

PRODUCTION CONTROL. Weiterhin können Anforderungen an den Produktionsprozess gestellt werden (bspw. maximale Dauer für die Serienproduktion einer Komponente, Yield-Rate, etc.). Anforderungen dieser Art werden durch eine Untersuchung des Produktionsprozesses überprüft.

Im Gegensatz zum Objekttyp können einer Anforderung mehrere Verifikationsmethoden zugewiesen werden. Ist eine Anforderung mit mehr als einer Verifikationsmethode versehen, bedeutet dies, dass sich potenziell alle zugewiesenen Verifikationsmethoden zur Überprüfung der Erfüllung der Anforderung eignen. Grundsätzlich wird

jedoch nur eine Verifikationsmethode zur Verifikation von Anforderungen verwendet. Die tatsächlich verwendete Verifikationsmethode wird in einem späteren Entwicklungsschritt ausgewählt. Werden einer Anforderung mehrere Verifikationsmethoden zugewiesen, kann dies auch bedeuten, dass die Anforderung noch nicht spezifisch genug ist oder mehrere Eigenschaften spezifiziert, die jeweils durch andere Verifikationsmethoden verifiziert werden sollten.

Relevanz des PVM-Attributs im Anforderungsmanagement

Durch die korrekte Auswahl des PVM-Attributs während der Erstellung von Anforderungsdokumenten wird insbesondere die Erstellung von Testspezifikationen vereinfacht, da durch das PVM-Attribut bereits alle für die Testspezifikation relevanten Anforderungen markiert sind. Das PVM-Attribut ist auch für das Berichtswesen relevant, da dieses Attribut in Verbindung mit Trace-Links zu Testfällen zur Einschätzung der Testabdeckung verwendet wird.

Das Attribut wird genauso wie auch das Attribut Objekttyp in Reviews manuell überprüft. In diesem Review untersuchen RE-Experten, ob die einer Anforderung zugewiesenen empfohlenen Verifikationsmethoden angemessen sind und nehmen ggf. Änderungen vor. Weiterhin muss zwingend sichergestellt werden, dass dieses Attribut für alle sicherheitsrelevanten Anforderungen gesetzt ist.

7.2 KONSTRUKTION EINES KLASSIFIKATORS

Wie bereits in Kapitel 4 wird auch hier der *Knowledge Discovery in Databases Process* [75] verwendet. Da das Vorgehen grundsätzlich ähnlich ist, werden in diesem Abschnitt lediglich die Unterschiede zur Objekttypklassifikation genauer erläutert.

Datensatzerstellung

Zur Konstruktion des Datensatzes wurden dieselben Anforderungsdokumente verwendet, die auch bei der Objekttypklassifikation zum Einsatz kamen. Es handelt sich dabei zum großen Teil um System- und Komponentenlastenhefte der Fahrzeugaußen- und Innenraumelektronik, der Fahrerassistenzsysteme und der Telematik. Es wurde lediglich ein deutscher Datensatz angefertigt, da zum Zeitpunkt der Erstellung noch nicht ausreichend englischsprachige Anforderungen zur Verfügung standen. Für den Datensatz wurden zunächst alle deutschsprachige und als Anforderung markierte Objekte ausgewählt.

Vorverarbeitung

Auf den Datensatz wurden zur Verbesserung der Datensatzqualität ähnliche Vorverarbeitungsschritte angewendet, die auch auf die Datensätze der Objekttypklassifikation angewendet wurden. Zunächst wurden alle Anforderungen entfernt, die aus einer Vorlage übernommen wurden, da für diese Anforderungen die Verifikationsmethode bereits in der Vorlage festgelegt ist und der Klassifikator die korrekte Klassifikation dieser Anforderungen damit nicht beherrschen muss.

Duplikate wurden aus dem Datensatz entfernt, da oft duplizierte Objekte zu Overfitting des Klassifikators auf den Merkmalen dieser

VERIFIKATIONSMETHODE	ANZAHL ANFORDERUNGEN	RATIO
Systemtest	23.529	87,25%
Review	3.483	12,91%
Simulation/Analysis	646	2,40%
Formal Verification	832	3,09%
Process Audit	788	2,92%
Production Control	289	1,07%
Anzahl Anforderungen	26.966	
Klassenkardinalität	1,096	

Tabelle 7.1: PVM-Datensatz: Überblick

Objekte führt. Weiterhin verfälschen duplizieren Objekte, die sowohl in der Trainingsmenge als auch in der Testmenge enthalten sind die Evaluationsergebnisse.

Auch auf diesem Datensatz wurde die Reifegradfilterung durchgeführt: Alle Anforderungen, deren Reifegrad nicht auf *abgestimmt* gesetzt ist, wurden entfernt. Zusätzlich wurden alle Anforderungen entfernt, bei denen keine empfohlene Verifikationsmethode gesetzt wurde.

Auf die verbliebenen Anforderungen wurden die in Abschnitt 4.3 beschriebenen vier Vorverarbeitungsschritte angewandt: Zahlen wurden durch Nullen ersetzt, Bezeichner von Signalen, Systemen und Komponenten wurden durch den allgemeinen Bezeichner *IDENTIFIER* ersetzt, Abkürzungen wurden zur Vereinheitlichung ausgeschrieben und alle Buchstaben wurden in Kleinbuchstaben umgewandelt.

Überblick über den PVM-Datensatz

Tabelle 7.1 enthält einen Überblick über den Datensatz. Insgesamt stehen 26.966 Anforderungen zur Verfügung. Die am häufigsten verwendete Verifikationsmethode ist *Systemtest*, da ein Großteil der Anforderungen in den Spezifikationen Systemeigenschaften und Systemverhalten spezifizieren, welches am implementierten System sichtbar und verifizierbar ist. Alle weiteren Verifikationsmethoden wurden nur spärlich verwendet. Die Klassenkardinalität [18] gibt an, wie viele Klassen einem Objekt im Datensatz durchschnittlich zugeordnet sind. In diesem Datensatz wurde fast allen Anforderungen im Datensatz genau eine Verifikationsmethode zugeordnet.

Die empfohlenen Verifikationsmethoden sind voneinander abhängig. Bestimmte Verifikationsmethoden werden häufig zusammen mit anderen Verifikationsmethoden verwendet. Tabelle 7.2 zeigt die Abhängigkeiten zwischen den Verifikationsmethoden und zeigt, wie wahrscheinlich eine bestimmte Verifikationsmethode zusammen mit einer anderen Verifikationsmethode verwendet wird. Diese Analyse zeigt, dass zum Beispiel für fast alle Anforderungen (95%), für welche die Verifikationsmethode *Simulation/Analysis* empfohlen wird,

WENN \ DANN	SIMULATION/ANALYSIS	SYSTEMTEST	REVIEW	PRODUCTION CONTROL	PROCESS AUDIT	FORMAL VERIFICATION
Simulation/Analysis	-	0,95	0,03	0,00	0,00	0,00
Systemtest	0,03	-	0,02	0,00	0,01	0,01
Review	0,01	0,16	-	0,08	0,19	0,13
Production Control	0,00	0,03	0,95	-	0,11	0,25
Process Audit	0,00	0,43	0,86	0,04	-	0,43
Formal Verification	0,00	0,40	0,53	0,09	0,40	-

Tabelle 7.2: Abhängigkeiten zwischen Verifikationsmethoden

auch die Verifikationsmethode *Systemtest* vorgeschlagen wird. In umgekehrter Richtung ist dies nicht der Fall (3%). Offensichtlich wird *Simulation/Analysis* für bestimmte Anforderungen als Alternative zum ebenfalls möglichen Systemtest vorgeschlagen.

Dieser Zusammenhang besteht auch zwischen den Verifikationsmethoden *Production Control* und *Review* (95%), sowie *Process Audit* und *Review*. Hier liegt die Vermutung nahe, dass Prozess- und Produktionsvorgaben nicht nur durch eine direkte Überprüfung der Prozesse, sondern auch durch ein Review von Dokumenten verifiziert werden können, die diese Prozesse beschreiben.

Die Verifikationsmethode *Formal Verification* wird oftmals zusammen mit anderen Verifikationsmethoden verwendet. Formale Überprüfungen von Systemeigenschaften können sehr zeitaufwändig sein und daher werden für Anforderungen, für die diese Verifikationsmethode vorgeschlagen wird, alternative und möglicherweise zeiteffizientere Verifikationsmethoden empfohlen.

Konstruktion eines Klassifikators

Zur Klassifikation des PVM-Attributs von Anforderungen wird wiederum ein Convolutional-Neural-Network erstellt und trainiert, da dieses Klassifikationsmodell bei der Objekttypklassifikation die besten Ergebnisse erzielt hat. Bei der Klassifikation der empfohlenen Verifikationsmethode können einer Anforderung jedoch mehrere Verifikationsmethoden zugeordnet werden, daher handelt es sich um ein *Multilabel*-Klassifikationsproblem. Die Architektur des Netzwerks musste dazu leicht angepasst werden.

Als letzte Operation des CNNs wird für die Objekttypklassifikation sie Softmax-Operation angewendet. Diese wandelt die Ausgabe aller

Netzwerkneuronen der Ausgabeebene in eine Wahrscheinlichkeitsverteilung um und stellt damit sicher, dass die Summe aller Netzwerkausgaben 1 ist und alle individuellen Netzwerkausgaben zwischen 0 und 1 liegen. Diese Operation ist für die Objekttypklassifikation geeignet, da jedem Objekt einer Spezifikation genau ein Objekttyp zugewiesen werden soll. Zur Multilabel-Klassifikation wird stattdessen die *Sigmoid*-Funktion verwendet, die jede Netzwerkausgabe unabhängig von allen anderen Ausgaben in eine Wahrscheinlichkeit zwischen 0 und 1 umwandelt. So kann das Netzwerk für mehrere Klassen eine hohe Wahrscheinlichkeit ausgeben.

Training des Klassifikators

Das Netzwerk wurde mittels zehnfacher Kreuzvalidierung trainiert und getestet. Die Unterteilung des Datensatzes in die zehn Teilmengen wurde zufällig durchgeführt. Dadurch wird eine ungefähre Gleichverteilung der einzelnen Klassen in den Teilmengen gewährleistet. Der Einsatz von *Stratification* war nicht möglich, da es sich bei der PVM-Klassifikation um ein Multilabel-Problem handelt [100]. Das Word2Vec-Modell wurde auf der gleichen Datenmenge trainiert, die auch für das Training der Word2Vec-Modelle der Objekttypklassifikation verwendet wurde (siehe Abschnitt 4.4).

Weiterhin ist der Datensatz nicht ausbalanciert. Um zu verhindern, dass ein Klassifikator jeder zu klassifizierenden Anforderung die Verifikationsmethode *Systemtest* zuweist, müssen während des Trainings Gegenmaßnahmen getroffen werden. Für den Ausgleich der Klassen wurden Klassengewichte verwendet. Diese geben während des Trainings den unterrepräsentierten Klassen mehr Gewicht und sorgen bei diesen Klassen für proportional größere Änderungen an den Gewichten des Netzwerks.

7.3 EVALUATION

Zur Bewertung der Leistung des Klassifikators wurde zuerst die Leistung eines ZeroR-Klassifikators auf dem Datensatz bestimmt. Dieser Klassifikator weist jeder Anforderung die im Datensatz am häufigsten vorkommende Klasse (Verifikationsmethode *Systemtest*) zu, ohne auf die Merkmale der jeweiligen Anforderung zu achten. Jeder beliebige andere Klassifikator, der mittels des Datensatzes trainiert wird, sollte besser sein als ZeroR.

Daraufhin wurden unterschiedliche Konfigurationen der Hyperparameter Word-Embedding-Größe, Filterlängen und Filteranzahl je Länge erstellt und trainiert, um die für dieses Klassifikationsproblem beste Konfiguration des Netzwerks zu ermitteln. Dieses Vorgehen wird *Grid-Search* genannt [17]. Auch wurde getestet, ob trainierbare Word-Embeddings zu besseren Ergebnissen führen.

Metriken zur Auswertung des Multilabel-Klassifikators

Für alle Konfigurationen des CNNs und für den ZeroR-Klassifikator werden die folgenden Metriken verglichen:

GENAUIGKEIT. Multilabel-Genauigkeit [102]. Diese Metrik berücksichtigt auch teilweise korrekt klassifizierte Objekte, bei denen der Klassifikator einige, aber nicht alle Klassen eines Beispiels erkannt hat oder dem Beispiel zu viele Klassen zugewiesen hat. Multilabel-Genauigkeit ist definiert als

$$\text{Multilabel-Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|} \tag{7.1}$$

, wobei n die Anzahl an Beispielen im Testdatensatz, Y_x die Menge der im Datensatz für das Beispiel x definierten Klassen und Z_x die Menge der vom Klassifikator für das Beispiel x vorhergesagten Klassen ist.

PERFECT-MATCH-RATIO. Diese Metrik berücksichtigt keine Objekte, die nur teilweise richtig klassifiziert sind, und zählt lediglich, bei wie vielen Beispielen im Testdatensatz jeweils alle Klassen korrekt zugeordnet wurden. Perfect-Match-Ratio ist definiert als

$$\text{Perfect-Match-Ratio} = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = Z_i) \tag{7.2}$$

, wobei die Indikator-Funktion I für den Fall $Y_i = Z_i$ 1 zurückgibt und ansonsten 0.

MAKRO-F_1. Der Makro-Durchschnitt des F_1-Werts des Klassifikators auf allen Klassen. Der Durchschnitt wird gebildet, indem zuerst der F_1-Wert des Klassifikators auf jeder individuellen Klasse bestimmt und daraufhin der Durchschnitt dieser Werte berechnet wird.

MIKRO-F_1. Der Mikro-Durchschnitt des F_1-Werts des Klassifikators. Dieser Wert wird als Durchschnitt aller Beispiele des Testdatensatzes ohne vorherige Bildung eines F_1-Wertes je Klasse gebildet.

Generell gilt, dass Makro-F_1 stärker durch unterrepräsentierte Klassen beeinflusst wird, während Mikro-F_1 hauptsächlich durch die Klassen mit den meisten Beispielen beeinflusst wird [102, 105]. Daher ist für die Bewertung der Leistung des Klassifikators besonders Makro-F_1 von Bedeutung.

Auswertung der Grid-Search-Ergebnisse

Die Ergebnisse des ZeroR-Klassifikators und aller getesteten Konfigurationen befinden sich in Tabelle 7.3. Der ZeroR-Klassifikator hat wie zu erwarten hohe Werte für die Metriken Genauigkeit, Perfect Match Ratio und Mikro-F_1 erreicht. Ein guter Klassifikator sollte diese Werte deutlich überbieten.

EMBEDDING-GRÖSSE	TRAINIERBARE EMBEDDINGS	FILTERLÄNGE	FILTERANZAHL	GENAUIGKEIT	PERFECT-MATCH-RATIO	MAKRO-F_1	MIKRO-F_1
ZeroR Baseline				0,8477	0,8293	0,1553	0,8323
32	Nein	2, 3	8	0,8979	0,8682	0,5818	0,9003
32	Nein	2, 3	64	0,9099	0,8824	0,7131	0,9153
32	Nein	2, 3, 4	8	0,9016	0,8718	0,6119	0,9042
32	Nein	2, 3, 4	64	0,9177	0,8930	0,7331	0,9226
32	Ja	2, 3	8	0,9130	0,8856	0,7127	0,9177
32	Ja	2, 3	64	0,9266	0,9044	0,7644	0,9316
32	Ja	2, 3, 4	8	0,9140	0,8872	0,7179	0,9177
32	Ja	2, 3, 4	64	0,9275	0,9070	0,7706	0,9331
128	Nein	2, 3	8	0,9057	0,8766	0,6705	0,9098
128	Nein	2, 3	64	0,9234	0,9019	0,7641	0,9286
128	Nein	2, 3, 4	8	0,9055	0,8768	0,6954	0,9104
128	Nein	2, 3, 4	64	0,9249	0,9048	0,7696	0,9314
128	Ja	2, 3	8	0,9220	0,8986	0,7523	0,9265
128	Ja	2, 3	64	**0,9313**	**0,9122**	0,7809	0,9361
128	Ja	2, 3, 4	8	0,9260	0,9049	0,7623	0,9307
128	Ja	2, 3, 4	64	0,9304	0,9119	0,7549	0,9360
256	Nein	2, 3	8	0,9082	0,8800	0,6948	0,9121
256	Nein	2, 3	64	0,9274	0,9067	0,7753	0,9332
256	Nein	2, 3, 4	8	0,9130	0,8876	0,7157	0,9173
256	Nein	2, 3, 4	64	0,9270	0,9068	0,7707	0,9329
256	Ja	2, 3	8	0,9224	0,9002	0,7518	0,9281
256	Ja	2, 3	64	0,9310	0,9115	**0,7904**	**0,9368**
256	Ja	2, 3, 4	8	0,9251	0,9043	0,7671	0,9307
256	Ja	2, 3, 4	64	0,9261	0,8995	0,6960	0,9308

Tabelle 7.3: Ergebnisse des PVM-Klassifikators mit unterschiedlichen Konfigurationen

VERIFIKATIONSMETHODE	SUPPORT	PRECISION	RECALL	F_1
Systemtest	23.529	0,970	0,977	0,973
Review	3.483	0,818	0,800	0,809
Simulation/Analysis	646	0,755	0,563	0,645
Formal Verification	832	0,910	0,692	0,786
Process Audit	788	0,865	0,787	0,824
Production Control	289	0,856	0,599	0,705
Mikro-Durchschnitt		0,944	0,930	0,937
Makro-Durchschnitt		0,862	0,736	0,790

Tabelle 7.4: Detaillierte Ergebnisse des PVM-Klassifikators

Insgesamt führen mehr Filter, größere Word-Embeddings und trainierbare Word-Embeddings zu besseren Klassifikationsergebnissen. Die Anzahl der Filter je Länge hat den größten Einfluss auf die Leistung des Netzwerks, da es dem Netzwerk durch eine größere Anzahl an Filtern möglich ist, mehr Wortmuster zur Klassifizierung zu erlernen. Auch die Erhöhung der Embedding-Größe von 32 auf 128 führt zu besseren Klassifikationsergebnissen. Wenn die Größe der Embeddings klein ist (32), führen trainierbare Word-Embeddings zu besseren Ergebnissen. Bei größeren Word-Embeddings ist in den gesammelten Ergebnissen keine Tendenz bei trainierbaren Word-Embeddings erkennbar. Die zusätzliche Verwendung von Filtern der Länge 4 hat keinen Einfluss auf die Leistung des Netzwerks. Diese Beobachtungen decken sich größtenteils mit den Beobachtungen aus Abschnitt 6.1, die mittels Random-Search auf der Objekttypklassifikation gemacht wurden.

Auswertung der besten Konfiguration

Um die Genauigkeit des Klassifikators besser beurteilen zu können, zeigt Tabelle 7.4 Precision, Recall und F_1 aller Klassen für die Netzwerkkonfiguration mit dem größten Makro-F_1-Wert (Größe der Word-Embeddings 256, trainierbare Word-Embeddings, Filter der Längen 2 und 3, 64 Filter je Länge).

Der Klassifikator erreicht für die Klasse *Systemtest* insgesamt 97% Recall und Precision. Der ZeroR-Klassifikator erreicht für diese Klasse 100% Recall und 87,25% Precision (vgl. Tabelle 7.1). Dieses Ergebnis zeigt, dass der Klassifikator sehr gut in der Lage ist, Anforderungen, die durch einen Systemtest verifiziert werden sollen, von Anforderungen zu unterscheiden, die mittels einer anderen Verifikationsmethode verifiziert werden sollen. Der Klassifikator liefert für die Klassen *Review* und *Process Audit* akzeptable Ergebnisse. Insbesondere Recall ist für die verbleibenden Verifikationsmethoden *Simulation/Analysis*, *Formal Verification* und *Production Control* sehr gering. Der Klassifikator ist bei fast der Hälfte der Beispiele dieser Verifikationsmethoden nicht in der Lage, die Verifikationsmethode korrekt zu identifizieren.

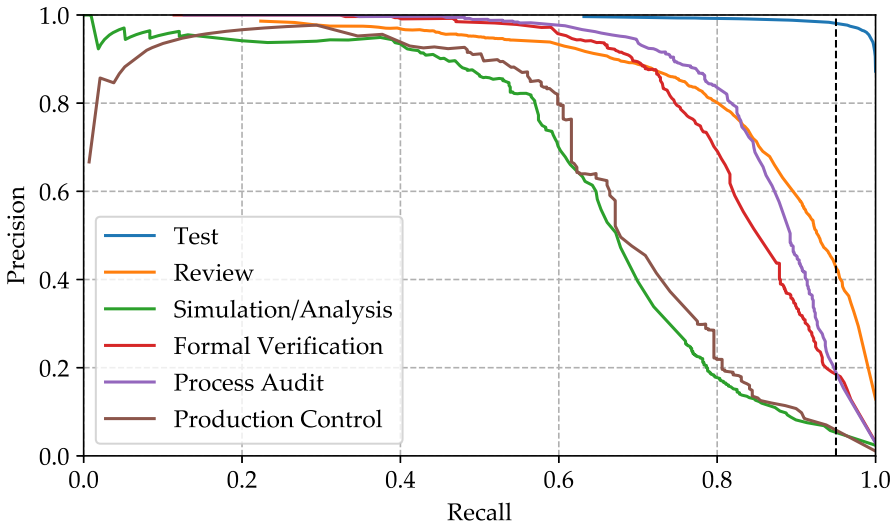


Abbildung 7.1: Multilabel-Klassifikation: Recall-Precision-Graph

Da Recall in der Regel wichtiger ist als Precision [20], kann die Abstimmung des Klassifikators auf höheren Recall und niedrigere Precision von Vorteil sein. Dazu wurden für alle Klassen Recall-Precision-Graphen erstellt. Abbildung 7.1 zeigt diese Graphen. Bei Erhöhung des Recalls auf 95% durch die Wahl eines geeigneten Schwellenwerts sinkt die Precision von fünf der sechs Klassen unter 50%, teilweise sogar unter 10%. Für eine Verwendung des Klassifikators in einem Werkzeug analog zur Objekttypklassifikation sind diese Ergebnisse nicht ausreichend.

7.4 ANALYSE EINFLUSSREICHER WÖRTER

Zusätzlich zur quantitativen Evaluation anhand der Metriken Precision, Recall und F_1 wurde eine qualitative Untersuchung des Klassifikators durchgeführt. Dabei wurde der in Abschnitt 5.2 beschriebene Ansatz zur Visualisierung der Klassifikationseinflüsse verwendet. Dieser Ansatz untersucht, welche Wörter einer Anforderung besonders relevant für die Klassifikationsentscheidung sind. Tabelle 7.5 zeigt für jede Verifikationsmethode einige Beispielanforderungen und die mit dem Ansatz generierten Visualisierungen der relevanten Wörter.

Der Einfluss einzelner Wörter hängt aufgrund der Beschaffenheit des CNNs stets von den Wörtern in näherer Umgebung ab. Aus diesem Grund haben beispielsweise die beiden Vorkommen des Worts „parametrierbare“ jeweils unterschiedlichen Einfluss auf die Klassifikationsentscheidung. Die Visualisierungen basieren auf den vom Klassifikator erlernten Merkmalen der jeweiligen Klassen. Daher ist die Aussagekraft der Visualisierungen abhängig von der Genauigkeit des Klassifikators auf der jeweiligen Klasse.

SYSTEMTEST
Die Aktuatoren und die Bedienschalter müssen einzeln im Steuergerät aktiviert sein.
Die Funktion wird ausgelöst wenn für eine parametrierbare Zeit eine parametrierbare Regenstärke erkannt wird.
REVIEW
Der Auftragnehmer muss einen geeigneten Testaufbau und Testablauf mit dem Fachbereich abstimmen.
Bei der Verwendung eines Gitters vor den Ultraschallkapseln ist die Erfüllung der weltweiten Gesetzes- und Versicherungsanforderungen sicherzustellen.
Alle CAN-Signale müssen immer sinnvolle Werte aufweisen und sind mit dem Auftragnehmer abzustimmen.
PRODUCTION CONTROL
Alle Werkstoffe und Verfahren müssen den geltenden gesetzlichen Bestimmungen in Bezug auf geregelte Stoffe und Wiederverwertbarkeit entsprechen.
Das Gewicht der Fernbedienung muss im funktionsfähigen Zustand in einem Rahmen von 0 bis 0 g liegen.
FORMAL VERIFICATION
Die Konsistenz zwischen Type-1-RAM und Type-2-RAM ist sicherzustellen.
SIMULATION/ANALYSIS
Um die Einsatzfähigkeit der Bussysteme zu gewährleisten, dürfen die Bussysteme keinesfalls beeinträchtigt oder gar blockiert werden.
Da eine elektronische Fanfare eine Totzeit von zirka 0 ms nach Beginn der Anforderung hat, muss die Fanfare für mindestens 0 ms angesteuert werden.
PROCESS AUDIT
Der Auftragnehmer muss die Anforderungen bezüglich des Logistikkonzeptes gemäß der Beschreibung im separaten Logistik-KLH erfüllen.
Während des gesamten Entwicklungszeitraumes muss der Auftragnehmer gewichtsreduzierende Maßnahmen betrachten und aufzeigen.

Tabelle 7.5: Visualisierte Beispiele der PVM-Klassifikation

Simulation / Analysis

Die Verifikationsmethode *Simulation/Analysis* wird im Datensatz häufig für Anforderungen verwendet, die Funktionalität definieren, welche von dem Zustand des Zündschlosses abhängig ist oder konkrete Werte für Spannungen, Zeitdauern, Entfernungen oder Geschwindigkeiten definieren. Die zweite gezeigte Anforderung dieser Klasse spezifiziert beispielsweise Zeiten für eine elektronische Fanfare und die Zeiteinheit „ms“ ist besonders relevant für die Klassifikation. Generell sind alle Anforderungen, die Zahlen kombiniert mit Einheiten enthalten prädestiniert für diese Verifikationsmethode.

Process Audit

Die Verifikationsmethode *Process Audit* wird für Anforderungen verwendet, die Eigenschaften von Entwicklungsprozessen spezifizieren. Diese Anforderungen spezifizieren häufig Rechte und Pflichten von Auftragnehmer und Auftraggeber während der Entwicklung einer Komponente. Daher sind besonders diese Wörter relevant für die Zuordnung einer Anforderung zur Klasse *Process Audit* und werden in den beiden Beispielanforderungen besonders stark hervorgehoben.

Production Control

Production Control wird für Anforderungen verwendet, die Vorgaben für die Produktion einer Komponente spezifizieren. Dazu gehören unter anderen Terminvorgaben, Materialvorgaben und Gewichtsvorgaben. Dementsprechend sind Wörter wie „Werkstoffe“, „Verfahren“, „Gewicht“ und Gewichtsspezifikationen („0 g liegen“) relevant für die Klassifikation dieser Verifikationsmethode.

Review

Wenn eine Anforderung keinen Systemtest benötigt und stattdessen durch ein manuelles Review verifiziert wird, wird diese Anforderung mit der Verifikationsmethode *Review* versehen. Meist handelt es sich dabei wie in den in der Tabelle gezeigten Beispielen um Anforderungen, die sich nur schwer durch einen automatisierten oder manuellen Testfall abdecken lassen. Werden die relevanten Wörter dieser Klasse mit denen der Klasse *Process Audit* verglichen, so fällt auf, dass ähnliche Wörter hervorgehoben werden (hier: „Auftragnehmer“). Dies liegt daran, dass die Verifikationsmethode *Review* sehr häufig zusammen mit der Verifikationsmethode *Process Audit* vergeben wird (siehe dazu Tabelle 7.2).

Systemtest

Die Verifikationsmethode *Systemtest* wird für nahezu alle Anforderungen verwendet, die sich nicht einer der anderen Verifikationsmethode zuordnen lassen. Daher werden hier Wörter hervorgehoben, die einen Satz als Anforderung charakterisieren („müssen aktiviert sein“, „wird ausgelöst wenn“).

Formal Verification

Zur Verifikationsmethode *Formal Verification* lassen sich keine Aussagen darüber treffen, welche Wörter oder Wortgruppen besonders relevant für diese Klasse sind. Ein Großteil der mit dieser Verifikationsmethode markierten Anforderungen unterscheiden sich nicht von den Anforderungen, die mit *Systemtest* markiert sind. Die gezeigte Beispielanforderung ist eine Anforderung, für die eine formale Verifikation angebracht ist, beispielsweise durch einen Beweis auf Basis der Eigenschaften der beiden Speicherarten und der verwendeten

Entwicklungsframeworks. Das hervorgehobene Wort „sicherzustellen“ ist dasjenige Wort in der Anforderung, welches noch am ehesten mit einem formalen Beweis assoziiert werden kann.

Insgesamt sind die Erkenntnisse durch den Einsatz der Visualisierungstechnik teilweise nützlich. Die schlechte Leistung des Klassifikators auf einigen Klassen im Datensatz und die damit verbundene schlechte Qualität der Visualisierungen dieser Klassen erschwert es, Aussagen über die für die jeweiligen Klassen relevanten Wörter zu treffen, da auch viele scheinbar nicht relevante Wörter hervorgehoben werden.

7.5 BINÄRE KLASSIFIKATION

Da die Leistung des Klassifikators nur auf der Klasse *Systemtest* akzeptable Ergebnisse lieferte, wurde ein weiterer Klassifikator trainiert, der eine binäre Klassifikation vornimmt und lediglich zwischen den beiden Klassen *Systemtest* und *Andere Methode* unterscheidet. Anforderungen, die durch einen *Systemtest* verifiziert werden, werden in den weiteren Entwicklungsprozessen grundsätzlich anders behandelt als Anforderungen, die mit anderen Verifikationsmethoden verifiziert werden. Insbesondere für die Erstellung von Testspezifikationen ist diese Unterscheidung relevant. Daher kann auch ein Klassifikator, der nur zwischen diesen beiden Klassen unterscheiden kann, hilfreich sein.

Konstruktion und Training des Klassifikators

Die den Beispielen im Datensatz zugewiesenen Verifikationsmethoden wurden wie folgt verändert: Alle Verifikationsmethoden abgesehen von *Systemtest* wurden entfernt. Alle Beispiele, denen nun keine Verifikationsmethode zugewiesen war, wurde die neue Klasse *Andere Methode* zugewiesen. Der Datensatz enthält damit weiterhin 26.966 Anforderungen, von denen 23.529 mit *Systemtest* und die verbleibenden 3.437 Anforderungen mit *Andere Methode* markiert sind. Das CNN wurde modifiziert und besitzt nun zwei Ausgabeneuronen und verwendet als letzte Netzwerkoperation wieder die Softmax-Operation statt der Sigmoid-Operation.

Das weitere Vorgehen ist größtenteils identisch zum Training des Multilabel-Klassifikators: Werte für die Hyperparameter wurden festgelegt, Klassengewichte zum Ausgleich des Ungleichgewichts der beiden Klassen eingesetzt und zehnfache Kreuzvalidierung zum Training und zur Bewertung des Klassifikators eingesetzt.

Auswertung der Ergebnisse

Tabelle 7.6 zeigt die Ergebnisse des Klassifikators. Recall ist für beide Klassen sehr gut, insbesondere werden fast alle Anforderungen erkannt, für die kein Systemtest notwendig ist. Analog dazu ist die Precision auf der Klasse *Systemtest* sehr gut: für beinahe alle Anforderungen, für die der Klassifikator die Klasse *Systemtest* bestimmt hat, ist auch ein Systemtest notwendig. Die niedrige Precision auf

VERIFIKATIONSMETHODE	PRECISION	RECALL	F_1
Systemtest (ZeroR)	0,872	1	0,932
Andere Methode (ZeroR)	-	0	-
Systemtest	0,997	0,969	0,983
Andere Methode	0,833	0,981	0,901
Mikro-Durchschnitt	0,975	0,971	0,972
Makro-Durchschnitt	0,916	0,975	0,942

Tabelle 7.6: Ergebnisse des binären Klassifikators

der Klasse *Andere Methode* zeigt, dass der Klassifikator derzeit diese Klasse favorisiert und diese im Zweifelsfall öfters zuweist als die Klasse *Systemtest*. Durch eine Änderung des Schwellenwerts kann dieses Verhalten nachträglich noch verändert werden.

7.6 ANWENDUNG UND DISKUSSION DER ERGEBNISSE

Durch die Ergebnisse ist schnell ersichtlich, dass der Einsatz des Multilabel-Klassifikators in einem Werkzeug analog zu dem in Abschnitt 6.2.1 gezeigten Werkzeug aufgrund der geringen Leistung auf einigen Klassen nicht praktikabel ist. Es ist jedoch möglich, den binären Klassifikator in einem solchen Werkzeug zur Unterstützung der korrekten Zuweisung von Anforderungen zur Verifikationsmethode *Systemtest* einzusetzen. Ein solches Werkzeug bietet RE-Experten gegenüber einem manuellen Review möglicherweise folgende Vorteile:

- Anwender des Werkzeugs können schnell Anforderungen identifizieren, für welche die Verifikationsmethode *Systemtest* nicht korrekt gesetzt ist (d.h., entweder fälschlicherweise nicht gesetzt oder fälschlicherweise gesetzt), da diese Anforderungen visuell hervorgehoben werden.
- Folglich können Anwender ebenfalls Anforderungen erkennen, bei denen die Zuweisung dieser Verifikationsmethode höchstwahrscheinlich korrekt ist und diese Anforderungen in einem Review überspringen. Dadurch kann analog zur Objekttypklassifikation während eines Reviews viel Zeit eingespart werden.
- Durch den Fokus auf Anforderungen mit wahrscheinlich falscher PVM-Klassifikation sollten Anwender durch den Einsatz des Werkzeugs eine genauere Klassifikation der Anforderungen hinsichtlich des PVM-Attributs erreichen können.

Die Vorteile und mögliche Risiken müssen allerdings erst mittels einer empirischen Studie analog zu den Experimenten in Abschnitt 6.2 und Abschnitt 6.3 untersucht werden. Da allerdings die Klassifikationaufgabe und das Anwendungsszenario ähnlich zur Objekttypklassifikation ist, können von ähnlichen empirischen Studien zur PVM-Klassifikation ebenfalls ähnliche Ergebnisse erwartet werden.

Da für den Großteil der Anforderungen die Verifikationsmethode *Systemtest* empfohlen wird ist die weitere Unterteilung dieser Verifikationsmethode ggf. sinnvoll. Tatsächlich existieren viele verschiedene Möglichkeiten, funktionale Anforderungen zu testen (Hardware-in-the-Loop, Tests im vollständigen System). Durch eine Unterteilung der Klasse *Systemtest* in beispielsweise die Klassen *Test (Komponente)*, *Test (Integration)* und *Test (Fahrzeug)* kann zum einen die Erstellung von Testspezifikationen und die Durchführung der Tests besser gesteuert werden. Zum anderen ermöglicht diese Unterteilung genaueres Berichtswesen.

Limitierungen der PVM-Klassifikation

Wie auch bei der Objekttypklassifikation birgt der Einsatz eines Werkzeugs zur Unterstützung der Verifikationsmethodenklassifikation einige Gefahren. Der Klassifikator wird niemals eine Genauigkeit von 100% erreichen können. Wenn es notwendig ist, 100% Genauigkeit zu erreichen, sollte vom Einsatz eines automatisierten Werkzeugs abgesehen werden. Allerdings ist es auch Menschen nicht möglich, diese 100% zu erreichen. Die Experimente zur Objekttypklassifikation haben nachgewiesen, dass es durch den Einsatz eines nicht perfekten Werkzeugs dennoch möglich ist, gegenüber manuellen Reviews bessere Ergebnisse zu erzielen.

Weiterhin handelt es sich bei den verwendeten sechs Verifikationsmethoden nicht um eine standardisierte Sammlung von Verifikationsmethoden, vielmehr wurden diese Verifikationsmethoden speziell für Anforderungsdokumente der Automobilindustrie erstellt. In Anforderungsdokumenten anderer Domänen werden möglicherweise andere Verifikationsmethoden verwendet oder möglicherweise auf die Dokumentation einer empfohlenen Verifikationsmethode je Anforderung komplett verzichtet. Das trainierte Modell ist daher nicht auf andere Domänen übertragbar, insbesondere auch, da es auf domänenspezifischen Anforderungen trainiert wurde und die Zuweisung von Verifikationsmethoden zu Anforderungen von den in der Automobilindustrie verwendeten Prozessen abhängig ist.

Eine weitere Limitierung ist, dass das Modell nach dem Training nicht mehr verändert werden kann. Sollten sich die Richtlinien zur Verwendung der Verifikationsmethoden ändern oder neue Verifikationsmethoden eingeführt werden, so muss das Modell erneut trainiert werden, um diese Änderungen zu berücksichtigen. Um diesem Problem entgegenzuwirken, können Active-Learning-Ansätze umgesetzt werden, durch die der Klassifikator während der Verwendung durch Anwender stetig aktualisiert wird [101]. Aufgrund von *Catastrophic*

Forgetting [95] ist dies allerdings keine einfach umzusetzende Aufgabe: Beim Training von neuronalen Netzwerken auf neuen Daten werden die Gewichte von Neuronen, die für die Klassifikation der ursprünglich trainierten Beispiele verantwortlich sind, schnell überschrieben und das Netzwerk verliert die Fähigkeit, diese Beispiele korrekt zu klassifizieren.

8 VERWANDTE ARBEITEN

In diesem Kapitel werden die mit dieser Dissertation verwandten Arbeiten vorgestellt und voneinander abgegrenzt.

8.1 QUALITÄTSSICHERUNG VON ANFORDERUNGS-SPEZIFIKATIONEN

Die in dieser Arbeit vorgestellte Klassifikation des Objekttyps von Anforderungen ist ein Ansatz zur Qualitätssteigerung von Anforderungsdokumenten. Da Anforderungsdokumente in vielen Entwicklungsschritten verwendet werden und eine hohe Qualität der Dokumente die effiziente und effektive Durchführung dieser Prozesse begünstigt, existieren viele Arbeiten, die auf unterschiedlichen Wegen die Qualität von Anforderungsdokumenten verbessern.

8.1.1 Boiler-Plates

Anforderungsspezifikationen werden häufig in natürlicher Sprache verfasst. Dies erleichtert das Verständnis und Bearbeitung dieser Dokumente, führt jedoch durch die großen Freiheitsgrade natürlicher Sprache oft zu Anforderungen, die grundlegende Qualitätskriterien wie Eindeutigkeit, Atomarität und Konsistenz [52] nicht erfüllen.

Boiler-Plates erzwingen die Verwendung bestimmter Sprachmuster in Anforderungen und verhindern dadurch Formulierungen, die zu mehrdeutigen oder nicht atomaren Anforderungen führen. EARS [77, 78] ist ein Boiler-Plate-Framework, welches solche Satzmuster definiert (siehe Tabelle 8.1). Diese Muster reichen aus, um jede mögliche Anforderung an ein System zu formulieren.

Die folgende Anforderung wurde ohne Einschränkungen der englischen Sprache verfasst [45]:

> The display should go off when the user has the phone by her face during a call.

Die gleiche Anforderung kann mithilfe des Musters *State-Driven* wie folgt umgeschrieben werden:

> While a CDMA or VoLTE call is active and the device senses the user's face in _proximity to the display, the device shall turn off the display.

NAME DES MUSTERS	MUSTER
Ubiquitous	The <system | actor> shall <action> <object>
Event-Driven	WHEN <trigger> <optional precondition> the <system | actor> shall <action> <object>
State-Driven	WHILE <system state | actor state>, the <system | actor> shall <action> <object>
Unwanted Behavior	IF <unwanted state | unwanted event>, THEN the <system | actor> shall <action> <object>
Optional Feature	WHERE <feature is included>, the <system | actor> shall <action> <object>
Complex	(Kombinationen der anderen Muster)

Tabelle 8.1: Satzmuster nach EARS [78]

Durch die Verwendung des EARS-Musters ist diese Anforderung nun eindeutig, da das State-Driven-Muster die Bedeutung der einzelnen Bestandteile der Anforderung definiert.

Weitere Boiler-Plate-Frameworks existieren, die grundsätzlich die gleichen Ziele verfolgen, dafür aber andere Satzmuster definieren. Ein weiteres und häufig zitiertes Framework ist das Boiler-Plate-Framework nach Rupp [8, 91]

Arora u. a. [9] haben einen Ansatz entwickelt, der basierend auf Text-Chunking automatisch überprüfen kann, ob Anforderungen den von Boiler-Plates vorgegebenen Satzmustern entsprechen. Text-Chunking zerlegt Sätze mithilfe eine Textparsers in nicht überlappende Teilsätze, die daraufhin mit den Boiler-Plate-Mustern verglichen werden. Der Ansatz funktioniert für beliebe Boiler-Plate-Frameworks, demonstriert wird der Ansatz an EARS und dem Framework nach Rupp. Die automatische Überprüfung von Anforderungen macht es einfacher, die Einhaltung der Satzmuster des Boiler-Plate-Frameworks in den Anforderungen einer Spezifikation sicherzustellen.

8.1.2 Mustererkennung

Oftmals muss die Qualität bestehender Anforderungen überprüft werden. Dazu existieren viele unterschiedliche Ansätze.

Requirements-Smells

Ein entsprechender Ansatz untersucht Anforderungen auf *Requirements-Smells* [35–37]. In diesen Arbeiten werden mehrere Smells vorgestellt, die auf Basis von Industriestandards definiert wurden. Zu diesen Smells gehören unter anderem:

SUBJEKTIVE SPRACHE. Die Verwendung von nicht objektiven Wörtern sollte vermieden werden, da diese von verschiedenen Perso-

nen unterschiedlich ausgelegt werden können. Beispiele: *benutzerfreundlich*, *einfach zu benutzen*, *kosteneffizient*.

RELATIVE FORMULIERUNGEN. Relative Formulierungen sind Formulierungen, die bestimmte Teile eines Systems mit anderen Systemen vergleichen. Beispiele: *besser als*, *schneller als*.

SCHLUPFLÖCHER. In Anforderungen sollten keine Formulierungen verwendet werden, die es ermöglichen, Anforderungen oder Teile von Anforderungen zu ignorieren. Beispiele: *wenn möglich*, *sofern zutreffend*, *soweit erforderlich*.

Als Teil der Arbeiten wurde ein Werkzeug entwickelt, welches unter Verwendung von NLP-Methoden Anforderungsdokumente analysiert und den Autor auf gefundene Qualitätsdefizite hinweist. Es bleibt dem Autor überlassen, die gefundenen Fehler zu korrigieren oder zu ignorieren.

Weak-Words

Andere Arbeiten beschäftigen sich mit sogenannten *Weak-Words* [64, 65]. Zu Weak-Words zählen Wörter wie *lang*, *besonders*, *sicher*, *klein* und *schnell*. Diese führen in bestimmten Kontexten zu mehrdeutigen, unklaren oder unverständlichen Anforderungen. Ob ein Weak-Word ein Problem ist, hängt von der Formulierung einer Anforderung ab:

1. Die LED muss für einen langen Zyklus leuchten.
2. Die LED muss für einen langen Zyklus von 3 Sekunden leuchten.

Nur die erste Anforderung ist nicht vollständig, da aufgrund der Verwendung des Weak-Words „langen“ die Dauer des Zyklus unbestimmt ist. In den Arbeiten von Krisch u. a. wird daher ein Ansatz vorgestellt, bei dem ebenfalls der Kontext von Weak-Words analysiert wird. Problematische Weak-Words werden mithilfe manuell definierter Regeln identifiziert. Warnungen werden nur dann ausgegeben, wenn ein Weak-Word tatsächlich ein Problem darstellt. Wie auch bei Requirements-Smells werden keine automatischen Korrekturen durchgeführt.

Mustererkennung mit NLP

Rosadini u. a. [96] untersuchten gezielt die Tauglichkeit von NLP-Techniken zur Suche nach Qualitätsmängeln in Anforderungsdokumenten. Eingesetzt werden Tokenizer, Part-of-Speech-Tagger, Shallow-Parsing, Wörterbücher und JAPE-Regeln. Mit diesen Techniken wurde in einem Datensatz nach Mustern wie vagen Ausdrücken, Passivsätzen und fehlenden Einheiten und Referenzen gesucht. Die Erkenntnisse der Arbeit kann in folgenden Punkten zusammengefasst werden:

- Ansätze auf Basis von NLP neigen dazu, viele False-Positives zu generieren. Deshalb ist es empfehlenswert, Werkzeuge auf Basis von NLP innerhalb eines Konzerns zu entwickeln. So können die Werkzeuge auf die speziellen Bedürfnisse des Konzerns angepasst und die Anzahl der Fehlalarme reduziert werden.

- Aufgrund der Komplexität von natürlicher Sprache entstehen bei der Verwendung von NLP zwangsläufig False-Negatives. Dies sind meistens Fälle, in denen Kontextinformationen (zum Beispiel domänenspezifisches Wissen) zur Identifikation eines Fehlers notwendig ist.
- Die Genauigkeit der Erkennung von Fehlern kann höchstens so gut sein, wie die Genauigkeit der NLP-Techniken. Ein wie von Berry u. a. [19] für Werkzeuge geforderter Recall von 100% ist in der Regel nicht möglich.

Yang u. a. [114] verwenden NLP-Techniken zur Suche nach anaphorischen Mehrdeutigkeiten in Anforderungen. Eine anaphorische Mehrdeutigkeit besteht, wenn ein Satzteil einer Anforderung auf mehrere andere Satzteile verweist und aus der Satzstruktur nicht hervorgeht, auf welchen anderen Satzteil verwiesen wird. In der Arbeit werden mithilfe von Regeln auf Part-of-Speech-Markierungen Mehrdeutigkeiten identifiziert und durch Experten bestätigt. Daraufhin wird ein Klassifikator trainiert, der diese Mehrdeutigkeiten automatisch identifizieren kann. Die Autoren haben ein prototypisches Werkzeug entwickelt, welches Mehrdeutigkeiten in einem Anforderungsdokument hervorhebt.

8.2 MASCHINELLES LERNEN IM ANFORDERUNGSMANAGEMENT

Anforderungsdokumente sind meist in natürlicher Sprache geschrieben und die Menge an zur Verfügung stehenden Anforderungen ist innerhalb eines Unternehmens meistens sehr groß. Daher bietet es sich an, Techniken aus dem Bereich Maschinelles Lernen auf Anforderungsdokumente anzuwenden. In vielen Arbeiten wird versucht, Anforderungen in diverse Kategorien automatisch zu klassifizieren.

Klassifikation von Anforderungen

Cleland-Huang u. a. [27] klassifizieren Anforderungen in funktionale und nicht-funktionale Anforderungen. Dazu wird ein einfacher, auf Schlüsselwörtern basierender Ansatz verwendet. Bei nicht-funktionalen Anforderungen wird zwischen unterschiedlichen Kategorien wie Aussehen, Performanz, Verfügbarkeit und Sicherheit unterschieden. Der erzielte Recall ist sehr gut, allerdings werden auch viele False-Positives gefunden, die von potenziellen Anwendern manuell aussortiert werden müssen. Die Autoren prognostizieren, dass die Klassifikation RE-Experten beispielsweise bei der gezielten Suche nach allen nicht-funktionalen Anforderungen eines Typs behilflich sein kann.

Hayes, Li und Rahimi [47] klassifizieren mithilfe von Weka[1] Anforderungen ebenfalls in funktionale und nicht-funktionale Anforderungen.

1 http://www.cs.waikato.ac.nz/ml/weka/

Weiterhin wird eine Klassifikation in temporale und nicht-temporale Anforderungen gezeigt. Die Autoren haben den Klassifikator in das von denselben Autoren entwickelte Werkzeug *TraceLab* integriert und argumentieren, dass mit ML-Techniken ausgestattete Werkzeuge zur Unterstützung vieler im Anforderungsmanagement derzeit manuell durchgeführter Aufgaben eingesetzt werden können.

Auch Casamayor, Godoy und Campo [23] klassifizieren Anforderungen in funktionale und nicht-funktionale Anforderungen, verwenden allerdings einen Semi-Supervised-Ansatz zur Klassifikation. Neben einem Datensatz von bereits klassifizierten Anforderungen wird ein weitaus größerer Datensatz nicht klassifizierter Anforderungen ebenfalls zum Training des Klassifikators verwendet, wodurch dessen Genauigkeit gesteigert wird. Weiterhin wird die Klassifikation einiger automatisch ausgewählte Anforderungen manuell durchgeführt, um so die Genauigkeit des Klassifikators weiter zu steigern.

Weitere Arbeiten zur Klassifikation funktionaler und nicht-funktionaler Anforderungen existieren besonders im Kontext der 2017 gestellten Data-Challenge der Requirements-Engineering-Konferenz [29, 66]. Die Arbeit von Abad u. a. [1] konnte dabei durch den Einsatz vieler aufwändiger Vorverarbeitungsschritte besonders gute Ergebnisse erzielen. Eingesetzt wurden NLP-Techniken zur Markierung von Part-of-Speech-Tags, Entity-Tagging und Temporal-Tagging. Auf dem vorgegebenen Datensatz wurde eine Precision von 0,95 und ein Recall von 0,94 erreicht.

Knauss und Ott [61, 87] klassifizieren Anforderungen mithilfe gängiger Klassifikationsverfahren (Naïve-Bayes, Support-Vector-Machines) in Themen. Themen sind in deren Arbeiten Konzepte wie beispielsweise Zeit, Temperatur und Spannung. Ein Klassifikator wurde mit Daten trainiert, in denen Anforderungen bereits manuell diesen Themen zugeordnet wurden. Der trainierte Klassifikator ist in der Lage, die Anforderungen eines Anforderungsdokuments nach Themen zu gruppieren. Die Autoren argumentieren, dass eine derartige Gruppierung besonders zur Beschleunigung von Reviews sinnvoll ist, da ein RE-Experte eine Menge von Anforderungen des gleichen Themas schneller überprüfen könne als viele durchmischte Anforderungen.

Transfer Learning

Hey u. a. [50] stellen einen Ansatz vor, der die für Transfer-Learning eingesetzte Technik BERT[2] auf die Klassifikation von Anforderungen überträgt. Transfer-Learning versucht, die von einem Modell erlernten Muster auf andere, ähnliche Probleme zu übertragen. Im Kontext der Anforderungsklassifikation kann Transfer-Learning das Erlernen von domänenübergreifenden Modellen erlauben. Die Autoren zeigen, dass die Anwendung von BERT bei der Klassifikation von funktionalen und nicht-funktionalen Anforderungen signifikante Verbesserungen erzielt.

2 Bidirectional Encoder Representations from Transformers

Identifikation von Anforderungen

Abualhaija u. a. [2] konstruieren einen Klassifikator, der in Anforderungsdokumenten Anforderungen von Nicht-Anforderungen unterscheiden kann. Im Gegensatz zu dieser Arbeit konzentrieren sich die Autoren allerdings auf Freiformtext. Durch Bestimmung von Satzgrenzen im Freitext werden Kandidaten für die Klassifikation ermittelt. Für die Klassifikation selbst werden auf syntaktischen und semantischen Spracheigenschaften manuell Features definiert, die daraufhin mit einfachen Klassifikatoren wie Random-Forests und Support-Vector-Machines zur Klassifikation verwendet werden.

8.3 UNTERSTÜTZEN MANUELLER AUFGABEN

Techniken wie die Klassifikation von Anforderungen können genutzt werden, um Entwickler bei zeitaufwändigen und manuell durchzuführenden Aufgaben zu unterstützen. Dazu wurden im Umfeld Anforderungsmanagement analog zu den in Abschnitt 6.2 und 6.3 vorgestellten Experimenten bereits mehrere Studien durchgeführt.

Knauss u. a. [60] benutzen einen Klassifikator, um RE-Experten bei der Identifikation von Sicherheitsanforderungen zu unterstützen. Anhand von drei Fallbeispielen wird zuerst gezeigt, dass der Klassifikator in der Lage ist, Sicherheitsanforderungen zu identifizieren. Daraufhin wird argumentiert, dass dieser Klassifikator durch den Fokus auf höchstwahrscheinlich sicherheitsrelevante Anforderungen RE-Experten in der Identifikation dieser unterstützen kann und die den RE-Experten zur Verfügung stehende Zeit besser genutzt werden kann.

Baum und Schneider untersuchen gezielt Werkzeuge zur Unterstützung des Reviews von Änderungen an Quelltext [14]. Quelltextänderungen werden in der Regel durch weitere Entwickler überprüft, bevor diese übernommen werden. In der Arbeit wird unter anderem untersucht, ob der Umfang der im Review zu überprüfenden Änderungen durch den Einsatz von Werkzeugen reduziert werden kann, da dadurch Zeit eingespart werden kann und die Entwickler möglicherweise öfters auf Probleme aufmerksam werden. Eine entsprechende Fallstudie wurde durchgeführt [13]. Durch den Einsatz von Modellen, die den Reviewbedarf von Quelltextänderungen vorhersagen können, kann in dieser Fallstudie ein Review von 25% der Quelltextänderungen und 23% der dazugehörigen Java-Quelltextzeilen vermieden werden, während lediglich 1% der ansonsten anfallenden Reviewanmerkungen nicht abgedeckt werden. Genauso wie die in Abschnitt 6.3 durchgeführte Fallstudie zeigt diese Arbeit den positiven Effekt des Einsatzes von Werkzeugen zur Unterstützung manueller Aufgaben.

Tjong und Berry [107] haben ein Werkzeug entwickelt, das in Anforderungsdokumenten nach Mehrdeutigkeiten sucht. Dabei wurde gemäß den Forderungen von Berry u. a. [19] primär auf die Optimie-

rung des Recalls geachtet und Precision vernachlässigt. Das Werkzeug weist auf den untersuchten Mehrdeutigkeiten einen Recall von 100% und eine Precision von etwa 60% auf. In einer Fallstudie auf zwei verschiedenen Anforderungsdokumenten konnte gezeigt werden, dass durch den Einsatz des Werkzeugs etwa 25% der für ein manuelles Review benötigten Zeit eingespart werden kann und dabei ähnlich viele Mehrdeutigkeiten gefunden werden.

8.4 ERKLÄREN VON NEURONALEN NETZWERKEN

Neuronale Netzwerke sind in der Lage, in großen Datenmengen komplexe Muster zu erkennen. Mit zunehmender Größe der Netzwerke wird es schwieriger, die von einem Netzwerk getroffenen Entscheidungen nachzuvollziehen. Dies ist aus zwei Gründen problematisch: Zum einen ist es schwierig, nachzuvollziehen, warum die Genauigkeit der Klassifikationsmodelle unter Umständen hinter den Erwartungen zurückbleibt. Zum anderen ist es für Anwender unmöglich, die Ausgaben neuronaler Netzwerke nachzuvollziehen. Daher beschäftigen sich viele vergangene und aktuelle Arbeiten mit der Interpretation neuronaler Netzwerke.

Erklären mit Fuzzy-Logic

Versuche zur Erklärung neuronaler Netzwerke reichen soweit zurück wie neuronale Netzwerke selbst. Benítez, Castro und Requena [16] stellen einen Ansatz vor, die Entscheidungen einfacher Feed-Forward-Netzwerke durch Fuzzy-Logic-Regeln zu erklären. Dazu wird zuerst gezeigt, dass Feed-Forward-Netzwerke und auf Fuzzy-Logic basierende Regelsysteme äquivalent sind. In einem zweiten Schritt werden Feed-Forward-Netzwerke in besser interpretierbare Fuzzy-Logic-Regelsysteme überführt. Später stellen die Autoren eine verbesserte Version des Ansatzes vor, der nun auch in der Lage ist, Netzwerke mit mehr als nur einer versteckten Ebene zu erklären [24]. Da heutige neuronale Netzwerke deutlich komplexer sind, wurden Fuzzy-Logic-Ansätze nicht weiter verfolgt.

Erklären von Bildklassifikatoren

Bach u. a. [11] stellen einen Ansatz vor, der es in neuronalen Netzwerken erlaubt, Ursachen für Klassifikationsentscheidungen zu untersuchen („Layer-Wise Relevance Propagation“). In der Arbeit werden Feed-Forward-Netzwerke zur Klassifikation von MNIST-Ziffern und Convolutional-Neural-Networks zur Klassifikation von ImageNet[3]-Bildern verwendet. In beiden Beispielen wird gezeigt, dass jeweils markante Merkmale in den klassifizierten Beispielen hervorgehoben werden. Bei der Klassifikation von Bussen im ImageNet-Datensatz werden beispielsweise vermehrt Fenster und Türen von Bussen hervorgehoben und eher selten Merkmale des Bildhintergrunds.

3 Eine offene Datenbank mit derzeit 14 Millionen klassifizierten Bildern. `http://www.image-net.org/`

Zeiler und Fergus [117] verfolgen das gleiche Ziel, verwenden jedoch einen anderen Ansatz. Auch hier versuchen die Autoren, die Funktionsweise von CNNs bei der Klassifikation von Bildern visuell zu erklären. Dazu werden Deconvolutional-Neural-Networks [118] eingesetzt. Mit diesen Netzwerken ist es möglich, die Aktivierungen einzelner Filter bei der Klassifikation von Beispielen zu visualisieren. Dies zeigt, welche Teile des Netzwerks auf welche Teile der Bildeingabe reagieren. Mit dem gezeigten Ansatz ist es ebenfalls möglich, relevante Teile im Eingabebild hervorzuheben.

Auch Yosinski u. a. [116] beschäftigen sich mit der Visualisierung von Convolutional-Neural-Networks. In deren Arbeit liegt der Fokus jedoch auch der Visualisierung der erlernten Filter und der Untersuchung, wie die Visualisierungen der Filter zu den zu klassifizierenden Bildern passen. Bei einem auf dem ImageNet-Datensatz trainierten Netzwerk haben die Autoren beispielsweise herausgefunden, dass obwohl es im Datensatz keine Klasse für Gesichter gibt ein Teil des Netzwerks erlernt hat, Gesichter zu erkennen. Viele Klassen in dem Datensatz beschreiben Lebewesen mit Gesichtern (Personen, Tiere).

Erklären von Textklassifikatoren

Li u. a. [70] visualisieren verschiedene Teile von LSTM-Netzwerken, die zur Klassifikation von Text eingesetzt werden. In der Arbeit wurden Filmrezensionen in positive und negative Rezensionen klassifiziert (auch Sentiment-Analyse genannt). Der Ansatz visualisiert die Aktivierungen, die bei der Klassifikation einzelner Beispiele in den LSTM-Ebenen entstehen. In vielen Beispielen ist erkennbar, dass bei besonders positiven („good“, „like“, „interesting“) und besonders negativen („hate“, „bad“, „boring“) Wörtern starke Aktivierungen für die jeweiligen Klassen auftreten. Weiterhin wird demonstriert, dass durch Intensivierungen („I hate it so much“ statt „I hate it“) stärkere Aktivierungen im Netzwerk hervorgerufen werden.

Arras u. a. [10] stellen einen Ansatz zur Erklärung der Entscheidungen von zur Textklassifikation eingesetzter Convolutional-Neural-Networks vor. Die Erklärungen erfolgen für die gleiche Netzwerkarchitektur, die auch in dieser Arbeit verwendet wird. In der grundlegenden Herangehensweise ist der Ansatzes identisch mit dem in Abschnitt 5.2 vorgestellten Ansatz. Beide Ansätze wurden etwa zur gleichen Zeit unabhängig voneinander entwickelt. Anhand eines trainierten Netzwerkes und eines klassifizierten Beispiels wird der hier *Relevanz* genannte Einfluss von Eingabeneuronen auf Ausgabeneuronen ausgehend von der Netzwerkausgabe zurückgerechnet. An diversen Beispielen werden Visualisierungen gezeigt, die zu den in dieser Arbeit benutzten Visualisierungen ähnlich sind.

Weitere Erklärungen von Klassifikatoren

Weitere Arbeiten existieren, die die Entscheidungen von beliebigen Klassifikatoren erklären. Baehrens u. a. [12] zeigen, wie auf Basis des Gradienten eines Klassifikators während des Trainings Erklärungsvektoren erzeugt werden können. Diese Vektoren zeigen an, welche Veränderungen in den Eingabedaten zu Veränderungen in den Aus-

gabedaten führen. Ribeiro, Singh und Guestrin [94] versuchen, Erklärungen für beliebige Klassifikationsmodelle zu generieren. Dazu wird ein wahrscheinlichkeitsbasiertes Modell entwickelt, welches eine Annäherung des Klassifikationsmodells ist und über das die Erklärungen generiert werden. Der Ansatz wird anhand von Textklassifikation demonstriert.

Andere Arbeiten untersuchen Visualisierungen von Feed-Forward-Netzwerken durch farbliche Hervorhebung aktiver Neuronen [108] und die Bestimmung relevanter Wörter bei der Klassifikation von Text durch Naive-Bayes-Klassifikatoren [111].

9 ZUSAMMENFASSUNG UND AUSBLICK

Die Erfassung von Anforderungen ist einer der ersten Schritte in der Entwicklung von neuen Systemen. Dabei entstehen Anforderungsdokumente, die die Eigenschaften und das Verhalten des zu entwickelnden Systems spezifizieren. Viele weitere Entwicklungsschritte basieren auf den Anforderungsspezifikationen, daher ist das Anforderungsmanagement eine der wichtigsten Aktivitäten in der Systementwicklung. In der Automobilindustrie dienen diese Dokumente auch als rechtliche Grundlage. Zulieferer sind dazu verpflichtet, Komponenten exakt nach den Anforderungen zu konstruieren. Verhält sich eine Komponente im Einsatz nicht wie spezifiziert, wird der Zulieferer dafür haftbar gemacht. Wird erforderliche Funktionalität nicht spezifiziert, so sind spätere Änderungen an den Anforderungen um ein Vielfaches teurer.

Anforderungsdokumente enthalten neben Anforderungen weitere Inhalte, die nicht zur Systemfunktionalität gehören und nicht rechtlich verbindlich sind. Diese Inhalte (Beispiele, Erläuterungen, Referenzen, Abbildungen, etc.) sind jedoch für das bessere Verständnis der Anforderungen hilfreich und reduzieren unter anderen den Kommunikationsbedarf zwischen Stakeholdern. In Anforderungsdokumenten wird daher strikt nach Anforderungen und Informationen getrennt. Diese Trennung geschieht derzeit manuell und die Sicherstellung der korrekten Trennung durch Expertenreviews sorgt für erheblichen Mehraufwand während der Erstellung der Anforderungsdokumente.

9.1 ZUSAMMENFASSUNG

In dieser Arbeit wurde ein Ansatz vorgestellt und evaluiert, der RE-Experten bei dieser manuellen Aufgabe unterstützt. Die Grundidee ist es, den Fokus der RE-Experten auf Spezifikationsobjekte zu lenken, die bezüglich des Objekttyps falsch klassifiziert sind. Dadurch sollen sowohl mehr Fehler in der Objekttypklassifikation gefunden werden als auch die für das Review benötigte Zeit reduziert werden.

Klassifikation von Anforderungen und Informationen

Zur Klassifikation von Spezifikationsobjekten in Anforderungen und Informationen wurde ein neuronales Netzwerk (Convolutional-Neural-Network) gemäß Kim [57] auf zwei Datenätzen trainiert, die jeweils aus deutschen und englischen Anforderungsdokumenten aus der Automobilindustrie erstellt wurden. Dieser Klassifikator erzielte

in einem Vergleich unterschiedlicher Klassifikationsmodelle die besten Ergebnisse auf den Datensätzen.

Erweiterungen

Der Klassifikator wurde gegenüber der von Kim vorgestellten Architektur erweitert und bezieht auch Kontextinformationen von Spezifikationsobjekten mit ein. Dadurch konnte die Genauigkeit des Klassifikators weiter gesteigert werden. Weiterhin wurde ein Ansatz entwickelt, der durch das Rückverfolgen von Berechnungen im Netzwerk die für eine Klassifikationsentscheidung relevanten Wörter in den Spezifikationsobjekten hervorhebt. Dieser Ansatz ist besonders hilfreich, um die vom Klassifikator getroffenen Entscheidungen einem Benutzer zu erklären.

Evaluation

Der Ansatz wurde zuerst mittels gängiger Methoden evaluiert. Durch Random-Search wurden die Hyperparameter des Klassifikators optimiert. Auf dem englischen Datensatz erreichte der Klassifikator damit eine Genauigkeit von 91%.

Zur weiteren Evaluation des Klassifikators wurde Dieser in ein Werkzeug integriert, welches in einer Spezifikation Warnungen bei falsch klassifizierten Objekten ausgibt. In einer Cross-over-Studie wurde daraufhin untersucht, ob der Einsatz des Werkzeugs in Reviews zu mehr korrigierten Fehlern und zu schneller durchführbaren Reviews führt. In einem Vergleich von zwei Gruppen, wobei eine Gruppe ohne Werkzeug und die andere Gruppe mit Werkzeug arbeitete, wurde gezeigt, dass durch den Werkzeugeinsatz die Anzahl der korrigierten Fehler steigt und die Anzahl neu eingeführter Fehler sinkt. Auch die Zeit verringert sich durch den Werkzeugeinsatz leicht. Insbesondere zeigt diese Studie allerdings auch, dass Fehler, die durch das Werkzeug nicht hervorgehoben werden, besonders häufig nicht entdeckt werden.

In einer zweiten empirischen Studie wurde daher gezielt die Relevanz von Recall gemäß Berry [20] untersucht. In diesem Experiment wurde das Werkzeug nicht verwendet, um Objekte mit Fehlern hervorzuheben, sondern um bereits korrekt klassifizierte Objekte aus dem Review auszuschließen. Die Ergebnisse der Studie zeigen, dass durch den Einsatz des Werkzeugs die Rate gefundener Fehler von durchschnittlich 39% auf 51% steigt, während die für das Review benötigte Zeit um 66% gesunken ist. Weitere Analysen zeigen, dass das Werkzeug in der Lage ist, die in einem Review zu untersuchenden Objekte um 61% zu reduzieren. Dies entspricht etwa einer Einsparung von 61% der in Reviews zur Überprüfung des Objekttyps verwendeten Zeit. Dabei wird sichergestellt, dass sich in den verbleibenden Objekten weiterhin durchschnittlich 98% aller Fehler befinden.

Klassifikation Empfohlene Verifikationsmethode

Der gleiche Ansatz wird daraufhin auf ein weiteres Klassifikationsproblem angewendet. Bei der Klassifikation von Anforderungen nach empfohlener Verifikationsmethode müssen Anforderungen zu insgesamt sechs Verifikationsmethoden zugeordnet werden. Auch dieses Attribut wird analog zum Objekttyp manuell gesetzt und in

Reviews überprüft. Diese Klassifikation lieferte unzureichende Ergebnisse, daher wurde das Problem auf ein binäres Klassifikationsproblem zwischen *Systemtest* und *Andere Testmethode* reduziert. Der Klassifikator lieferte auf diesem Problem mit F_1-Werten von 0.98 und 0.90 auf den beiden Klassen sehr gut Ergebnisse. Grundsätzlich ist der Ansatz damit auch auf andere Klassifikationsprobleme übertragbar und nicht nur auf das Objekttyp-Klassifikationsproblem beschränkt.

9.2 AUSBLICK

Insgesamt zeigen die in dieser Arbeit vorgestellten Ergebnisse großes Potential zur Unterstützung manueller Reviews. Dennoch existieren viele Punkte, an denen der Ansatz durch weitere Arbeiten verbessert und erweitert werden kann. Diese werden zum Abschluss der Arbeit erläutert.

Andere Klassifikationsmodelle

In dieser Arbeit wurden zur Klassifikation von Anforderungen Convolutional-Neural-Networks eingesetzt. Es existieren viele weitere Modelle, die zu anderen oder auch besseren Ergebnissen führen können [81]. *Recurrent-Neural-Networks* (RNN) sind Netzwerke, die Sequenzen von Eingabedaten verarbeiten können und eignen sich daher auch zur Verarbeitung von Text. *Long-Short-Term-Memory*-Netzwerke (LSTM) werden beispielsweise zur maschinellen Übersetzung eingesetzt, können aber auch zur Klassifikation verwendet werden. Weitere Arten neuronaler Netzwerke zur Klassifikation von Text existieren, unter anderem *Capsule-Neural-Networks* [51], *Attention-Networks* [115] und *Memory-Augmented-Networks* [84].

Durchführung weiterer Experimente

In dieser Arbeit wurden empirische Studien mit Studierenden durchgeführt. Da aufgrund der notwendigen Geheimhaltung der Experimentierdaten keine Online-Studien mit deutlich mehr Teilnehmern möglich war, war dies die beste Möglichkeit, möglichst viele Messergebnisse zu erhalten. Dennoch ist die Menge der Ergebnisse vergleichsweise klein und ist beispielsweise nicht groß genug, um statistische Signifikanztests durchzuführen. Lediglich die auf fast allen in den Experimenten verwendeten Dokumenten ähnlichen Ergebnisse untermauern die Aussagekraft der Ergebnisse. Weiterhin wurden keine Studien mit RE-Experten durchgeführt, da es hier noch weniger möglich war, genügend Teilnehmer mit ausreichend Zeit für die Studie zu gewinnen. Durch die Durchführung weiterer empirischer Experimente mit größeren Teilnehmergruppen können die Ergebnisse dieser Arbeit weiter bekräftigt werden.

Für die Klassifikation der empfohlenen Verifikationsmethode sind Ergebnisse einer Studie analog zu den zur Klassifikation des Objekttyps durchgeführten Studie interessant. Da es jedoch viele Ähnlichkeiten zur Objekttypklassifikation gibt, würde eine entsprechende Studie wahrscheinlich ähnliche Ergebnisse liefern.

Kombinierung Reduktion und Hervorhebung

In beiden Experimenten wurde das Review durch jeweils zwei unterschiedliche Methoden unterstützt. Im ersten Experiment wurden gezielt Objekte der Spezifikationen hervorgehoben, bei denen aufgrund der Klassifikation des Werkzeugs ein Fehler vermutet wurde. Im zweiten Experiment wurde gezeigt, dass durch den Ausschluss höchstwahrscheinlich richtig klassifizierter Objekte deutlich größere Verbesserungen besonders in der für die Reviews benötigten Zeit erzielt werden können. Beide Ansätze können kombiniert werden: Das Werkzeug kann sowohl Objekte verstecken als auch eindeutig falsch klassifizierte Objekte hervorheben. Damit ist es RE-Experten eventuell möglich, Reviews noch besser durchzuführen. Dies muss in weiteren Experimenten untersucht werden.

Active-Learning

Nachdem das Klassifikationsmodell des Werkzeugs trainiert wurde, ändern sich die Gewichte des Modells nicht mehr und die vom Werkzeug vorhergesagte Klassifikation eines Objekts wird sich nicht ändern. Gibt das Werkzeug wiederholt falsche Vorschläge, können RE-Experten von der Benutzung des Werkzeugs absehen. Durch *Active-Learning* werden die Gewichte von Klassifikationsmodellen auf Basis von Interaktionen mit Benutzern angepasst. Dadurch wäre das Werkzeug in der Lage, während der Benutzung fehlerhafte Vorschläge zu korrigieren und sich selbstständig an sich möglicherweise mit der Zeit ändernde Richtlinien bezüglich der Objekttypklassifikation anzupassen. Allerdings werden durch nachträgliche Änderungen an den Gewichten eines Netzwerks schnell viele erlernte Muster überschrieben. Dieser Effekt wird *Catastrophic-Forgetting* [95] genannt und es gibt umfassende Forschungsarbeiten, die versuchen diesem Effekt entgegenzuwirken [44, 56, 59].

Automatische Änderungen

Das Werkzeug führt keine automatischen Änderungen an Anforderungsdokumenten durch (beispielsweise vollständig automatische Korrektur eines Dokuments), da hierfür die Genauigkeit des Klassifikators nicht hoch genug ist. Dennoch gibt es Möglichkeiten, bestimmte Schritte zu automatisieren. Das Review von Anforderungsdokumenten ist insbesondere nötig, da während der Erstellung der Dokumente oftmals wichtige Attribute wie der Objekttyp nicht gesetzt werden. Ein Klassifikator kann beispielsweise eingesetzt werden, um bei neuen und geänderten Objekten bestimmte Attribute automatisch zu setzen. Durch scheinbar falsche Vorschläge des Werkzeugs werden Autoren möglicherweise auch auf lediglich schlecht formulierte Anforderungen aufmerksam gemacht.

Im ersten Experiment wurde untersucht, ob das Werkzeug Benutzer dazu verleitet, auch Änderungen am Text von Anforderungen und Informationen vorzunehmen. Es existieren Fälle, in denen das Werkzeug eine Änderung am Objekttyp vorschlägt, tatsächlich jedoch der Objekttyp korrekt ist und stattdessen das jeweilige Objekt präziser formuliert werden muss. Im Experiment wurden nur bei sehr wenigen Objekten Änderungen am Text aufgrund falscher Werkzeughinwei-

se durchgeführt. In weiteren Arbeiten kann untersucht werden, ob es mit Techniken des maschinellen Lernens möglich ist, Umformulierungsvorschläge für solche Objekte zu generieren und diese dem Anwender statt einer Änderung des Objekttyps vorzuschlagen. Zur automatischen Umformulierung von Anforderungen existieren bereits Arbeiten [103].

Anwendung in der Praxis

Wird das Werkzeug in der Praxis eingesetzt, sind insbesondere die folgenden zwei Punkte relevant. Anforderungsdokumente sind in der Regel in unterschiedliche Kapitel gegliedert (siehe Kapitel 3). In unterschiedlichen Kapiteln der Dokumente werden unterschiedliche Formulierungen verwendet. Daher ist es gegebenenfalls von Vorteil, je nach Kapitel ein anderes Klassifikationsmodell einzusetzen.

Zum anderen ist das trainierte Modell durch Training auf Anforderungsdokumenten, die lediglich in der Automobilindustrie stammen, domänenspezifisch und wird in anderen Domänen (Softwareentwicklung, Luft- und Raumfahrt, Bahn) unzureichende Ergebnisse liefern. Möglicherweise ist das Modell sogar spezifisch für den Konzern, auf dessen Anforderungsdokumenten das Modell trainiert wurde. Zum Einsatz in anderen Bereichen muss daher das Modell erneut trainiert werden.

LITERATUR

[1] Zahra Shakeri Hossein Abad, Oliver Karras, Parisa Ghazi, Martin Glinz, Guenther Ruhe und Kurt Schneider. „What Works Better? A Study of Classifying Requirements“. In: *Proceedings of the 25th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2017, S. 496–501. DOI: 10.1109/RE.2017.36.

[2] Sallam Abualhaija, Chetan Arora, Mehrdad Sabetzadeh, Lionel C. Briand und Eduardo Vaz. „A Machine Learning-Based Approach for Demarcating Requirements in Textual Specifications“. In: *Proceedings of the 27th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2019, S. 51–62. DOI: 10.1109/RE.2019.00017.

[3] Charu C. Aggarwal und ChengXiang Zhai. „A Survey of Text Classification Algorithms“. In: *Mining Text Data*. Hrsg. von Charu C. Aggarwal und ChengXiang Zhai. Springer US, 2012, S. 163–222. DOI: 10.1007/978-1-4614-3223-4_6.

[4] Charu C. Aggarwal und ChengXiang Zhai. „A Survey of Text Clustering Algorithms“. In: *Mining Text Data*. Hrsg. von Charu C. Aggarwal und ChengXiang Zhai. Springer US, 2012, S. 77–128. DOI: 10.1007/978-1-4614-3223-4_4.

[5] Saadia Ahmed und Hafiza Tehmins Kanwal. „Visualization Based Tools for Software Requirement Elicitation“. In: *Proceedings of the 2014 International Conference on Open Source Systems Technologies (ICOSST)*. 2014, S. 156–159. DOI: 10.1109/ICOSST.2014.7029337.

[6] Saleh Albelwi und Ausif Mahmood. „A Framework for Designing the Architectures of Deep Convolutional Neural Networks“. In: *Entropy* 19.6 (2017), S. 242–261. DOI: 10.3390/e19060242.

[7] I. Arel, C. Liu, T. Urbanik und A. G. Kohls. „Reinforcement learning-based multi-agent system for network traffic signal control“. In: *IET Intelligent Transport Systems* 4.2 (2010), S. 128–135. DOI: 10.1049/iet-its.2009.0070.

[8] Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand und Frank Zimmer. „Requirement Boilerplates: Transition from Manually-Enforced to Automatically-Verifiable Natural Language Patterns“. In: *Proceedings of the 4th IEEE International Workshop on Requirements Patterns (RePa)*. IEEE, 2014, S. 1–8. DOI: 10.1109/RePa.2014.6894837.

[9] Chetan Arora, Mehrdad Sabetzadeh, Lionel Briand und Frank Zimmer. „Automated Checking of Conformance to Requirements Templates Using Natural Language Processing“. In: *IEEE Transactions on Software Engineering* 41.10 (2015), S. 944–968. DOI: 10.1109/TSE.2015.2428709.

[10] Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller und Wojciech Samek. „What is relevant in a text document?: An interpretable machine learning approach“. In: *PLoS ONE* 12.8 (2017). DOI: 10.1371/journal.pone.0181142.

[11] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller und Wojciech Samek. „On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation“. In: *PLoS ONE* 10.7 (2015), S. 1–46. DOI: 10.1371/journal.pone.0130140.

[12] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen und Klaus-Robert Müller. „How to Explain Individual Classification Decisions“. In: *Journal of Machine Learning Research* 11 (2010), S. 1803–1831.

[13] Tobias Baum, Steffen Herbold und Kurt Schneider. „An Industrial Case Study on Shrinking Code Review Changesets through Remark Prediction“. In: *arXiv preprint* arXiv:1812.09510 (2018).

[14] Tobias Baum und Kurt Schneider. „On the Need for a New Generation of Code Review Tools“. In: *Product-Focused Software Process Improvement: 17th International Conference, PROFES*. Hrsg. von Pekka Abrahamsson, Andreas Jedlitschka, Anh Nguyen Duc, Michael Felderer, Sousuke Amasaki und Tommi Mikkonen. Springer International Publishing, 2016, S. 301–308. DOI: 10.1007/978-3-319-49094-6_19.

[15] Martin Beckmann, Andreas Vogelsang und Christian Reuter. „A Case Study on a Specification Approach Using Activity Diagrams in Requirements Documents“. In: *Proceedings of the 25th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2017, S. 253–262. DOI: 10.1109/RE.2017.28.

[16] José M. Benítez, Juan L. Castro und I. Requena. „Are Artificial Neural Networks Black Boxes?“ In: *IEEE Transactions on Neural Networks* 8.5 (1997), S. 1156–1164. DOI: 10.1109/72.623216.

[17] James Bergstra und Yoshua Bengio. „Random Search for Hyper-Parameter Optimization“. In: *Journal of Machine Learning Research* 13 (2012), S. 281–305.

[18] Flavia Cristina Bernardini, Rodrigo Barbosa da Silva, Rodrigo Magalhães Rodovalho und Edwin Benito Mitacc Meza. „Cardinality and Density Measures and Their Influence to Multi-

Label Learning Methods“. In: *Learning and Nonlinear Models* 12.1 (2014), S. 53–71. DOI: 10.21528/LNLM-vol12-no1-art4.

[19] Daniel Berry, Ricardo Gacitua, Pete Sawyer und Sri Fatimah Tjong. „The Case for Dumb Requirements Engineering Tools“. In: *Requirements Engineering: Foundation for Software Quality: 18th International Working Conference, REFSQ*. Hrsg. von Björn Regnell und Daniela Damian. Springer, 2012, S. 211–217. DOI: 10.1007/978-3-642-28714-5_18.

[20] Daniel M. Berry. „Evaluation of Tools for Hairy Requirements and Software Engineering Tasks“. In: *Proceedings of the 25th IEEE International Requirements Engineering Conference (RE), Workshops*. IEEE, 2017, S. 284–291. DOI: 10.1109/REW.2017.25.

[21] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 8. Aufl. 2009.

[22] Manfred Broy. „Challenges in Automotive Software Engineering“. In: *Proceedings of the 28th International Conference on Software Engineering (ICSE)*. ACM, 2006, S. 33–42. DOI: 10.1145/1134285.1134292.

[23] Agustin Casamayor, Daniela Godoy und Marcelo Campo. „Identification of non-functional requirements in textual specifications: A semi-supervised learning approach“. In: *Information and Software Technology* 52.4 (2010). DOI: 10.1016/j.infsof.2009.10.010.

[24] Juan L. Castro, Carlos J. Mantas und José M. Benítez. „Interpretation of Artificial Neural Networks by Means of Fuzzy Rules“. In: *IEEE Transactions on Neural Networks* 13.1 (2002), S. 101–116. DOI: 10.1109/72.977279.

[25] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall und W. Philip Kegelmeyer. „SMOTE: Synthetic Minority Over-sampling Technique“. In: *Journal of Artificial Intelligence Research* 16 (2002), S. 321–357.

[26] Dan Cireşan, Ueli Meier und Jürgen Schmidhuber. „Multi-column Deep Neural Networks for Image Classification“. In: *Proceedings of the 25th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, S. 3642–3649. DOI: 10.1109/CVPR.2012.6248110.

[27] Jane Cleland-Huang, Raffaella Settimi, Xuchang Zou und Peter Solc. „Automated classification of non-functional requirements“. In: *Requirements Engineering* 12.2 (2007), S. 103–120. DOI: 10.1007/s00766-007-0045-1.

[28] William W. Cohen und Yoram Singer. „Context-sensitive learning methods for text categorization“. In: *Proceedings of the 19th Annual International Conference on Research and Development in Information Retrieval (SIGIR)*. Association for Computing Machinery, 1996, S. 307–315. DOI: 10.1145/243199.243278.

[29] Alex Dekhtyar und Vivian Fong. „RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow“. In: *Proceedings of the 25th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2017, S. 484–489. DOI: 10.1109/RE.2017.26.

[30] Harris Drucker, Donghui Wu und Vladimir N. Vapnik. „Support Vector Machines for Spam Categorization“. In: *IEEE Transactions on Neural Networks* 10.5 (1999), S. 1048–1054. DOI: 10.1109/72.788645.

[31] John Duchi, Elad Hazan und Yoram Singer. „Adaptive Subgradient Methods for Online Learning and Stochastic Optimization“. In: *Journal of Machine Learning Research* 12 (2011), S. 2121–2159.

[32] Jonas Eckhardt, Andreas Vogelsang und Daniel Méndez Fernández. „On the Distinction of Functional and Quality Requirements in Practice“. In: *Product-Focused Software Process Improvement: 17th International Conference, PROFES*. Hrsg. von Pekka Abrahamsson, Andreas Jedlitschka, Anh Nguyen Duc, Michael Felderer, Sousuke Amasaki und Tommi Mikkonen. Springer International Publishing, 2016, S. 31–47. DOI: 10.1007/978-3-319-49094-6_3.

[33] Martin Ester, Hans-Peter Kriegel, Jörg Sander und Xiaowei Xu. „A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise“. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*. Hrsg. von Evangelos Simoudis, Jiawei Han und Usama Fayyad. AAAI Press, 1996, S. 226–231.

[34] Davide Falessi, Natalia Juristo, Claes Wohlin, Burak Turhan, Jürgen Münch, Andreas Jedlitschka und Markku Oivo. „Empirical software engineering experts on the use of students and professionals in experiments“. In: *Empirical Software Engineering* 23.1 (2018), S. 452–489. DOI: 10.1007/s10664-017-9523-3.

[35] Henning Femmer. „Reviewing Natural Language Requirements with Requirements Smells - A Research Proposal -“. In: *Proceedings of the 11th International Doctoral Symposium on Empirical Software Engineering (IDoESE)*. 2013.

[36] Henning Femmer, Daniel Méndez Fernández, Elmar Juergens, Michael Klose, Ilona Zimmer und Jörg Zimmer. „Rapid Requirements Checks with Requirements Smells: Two Case Studies“.

In: *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering (RCoSE)*. Hrsg. von Matthias Tichy, Jan Bosch, Michael Goedicke und Magnus Larsson. ACM, 2014, S. 10–19. DOI: 10.1145/2593812.2593817.

[37] Henning Femmer, Daniel Méndez Fernández, Stefan Wagner und Sebastian Eder. „Rapid quality assurance with Requirements Smells“. In: *Journal of Systems and Software* 123 (2017), S. 190–213. DOI: 10.1016/j.jss.2016.02.047.

[38] Daniel Méndez Fernández und Stefan Wagner. „Naming the pain in requirements engineering: A design for a global family of surveys and first results from Germany“. In: *Information and Software Technology* 57 (2015), S. 616–643. DOI: 10.1016/j.infsof.2014.05.008.

[39] Daniel Méndez Fernández, Stefan Wagner, Klaus Lochmann, Andrea Baumann und Holger de Carne. „Field study on requirements engineering: Investigation of artefacts, project parameters, and execution strategies“. In: *Information and Software Technology* 54.2 (2012), S. 162–178. DOI: 10.1016/j.infsof.2011.09.001.

[40] Juan M. Carrillo de Gea, Joaquín Nicolás, José L. Fernández Alemán, Ambrosio Toval, Christof Ebert und Aurora Vizcaíno. „Requirements engineering tools: Capabilities, survey and assessment“. In: *Information and Software Technology* 54.10 (2012), S. 1142–1157. DOI: 10.1016/j.infsof.2012.04.005.

[41] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter und Lalana Kagal. „Explaining Explanations: An Overview of Interpretability of Machine Learning“. In: *Proceedings of the 5th IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. Hrsg. von Francesco Bonchi, Foster Provost, Eliassi-Rad Tina, Wei Wang, Ciro Cattuto und Rayid Ghani. 2018, S. 80–89. DOI: 10.1109/DSAA.2018.00018.

[42] Yoav Goldberg. *Neural Network Methods for Natural Language Processing*. 10. Aufl. Synthesis Lectures on Human Language Technologies. 2017. DOI: 10.2200/S00762ED1V01Y201703HLT037.

[43] Ian J. Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. MIT Press, 2016.

[44] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville und Yoshua Bengio. „An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks“. In: *Proceedings of the 2014 International Conference on Learning Representations (ICLR)*. 2014.

[45] Sarah C. Gregory. *Writing Good Requirements, Version 7.0 09/16*. Tutorial at the 24th IEEE International Requirements Engineering Conference (RE), Beijing, China, 2016.

[46] Jane Huffman Hayes, Alex Dekhtyar und Senthil Karthikeyan Sundaram. „Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods“. In: *IEEE Transactions on Software Engineering* 32.1 (2006), S. 4–19. DOI: 10.1109/TSE.2006.3.

[47] Jane Huffman Hayes, Wenbin Li und Mona Rahimi. „Weka Meets TraceLab: Toward Convenient Classification: Machine Learning for Requirements Engineering Problems: A Position Paper“. In: *Proceedings of the 1st IEEE International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*. IEEE, 2014, S. 9–12. DOI: 10.1109/AIRE.2014.6894850.

[48] Simon Haykin. *Neural Networks and Learning Machines*. 3. Aufl. Pearson Education, 2009.

[49] Haibo He und Edwardo A. Garcia. „Learning from Imbalanced Data“. In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), S. 1263–1284. DOI: 10.1109/TKDE.2008.239.

[50] Tobias Hey, Jan Keim, Anne Koziolek und Walter F. Tichy. „NoRBERT: Transfer Learning for Requirements Classification“. In: *Proceedings of the 28th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2020. DOI: 10.1109/RE48521.2020.00028.

[51] Geoffrey E. Hinton, Alex Krizhevsky und Sida D. Wang. „Transforming Auto-Encoders“. In: *Artificial Neural Networks and Machine Learning: 21st International Conference on Artificial Neural Networks, ICANN*. Hrsg. von Timo Honkela, Włodzisław Duch, Mark Girolami und Samuel Kaski. Springer Berlin Heidelberg, 2011, S. 44–51. DOI: 10.1007/978-3-642-21735-7_6.

[52] ISO/IEC/IEEE 29148 International Standard. *Systems and software engineering – Life cycle processes –Requirements engineering*. 2011. DOI: 10.1109/IEEESTD.2011.6146379.

[53] Andreas Jedlitschka, Marcus Ciolkowski und Dietmar Pfahl. „Reporting Experiments in Software Engineering“. In: *Guide to Advanced Empirical Software Engineering*. Hrsg. von Forrest Shull, Janice Singer und Dag I. K. Sjøberg. Springer London, 2008, S. 201–228. DOI: 10.1007/978-1-84800-044-5_8.

[54] Thorsten Joachims. „Text Categorization with Suport Vector Machines: Learning with Many Relevant Features“. In: *Machine Learning: ECML-98. 10th European Conference on Machine Learning*. London, UK: Springer, 1998, S. 137–142. DOI: 10.1007/BFb0026683.

[55] Nal Kalchbrenner, Edward Grefenstette und Phil Blunsom. „A Convolutional Neural Network for Modelling Sentences“. In: *Proceedings of the 52nd Annual Meeting of the Association for*

Computational Linguistics (ACL). Association for Computational Linguistics, 2014, S. 655–665. DOI: 10.3115/v1/P14-1062.

[56] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler L. Hayes und Christopher Kanan. „Measuring Catastrophic Forgetting in Neural Networks“. In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 2018, S. 3390–3398.

[57] Yoon Kim. „Convolutional Neural Networks for Sentence Classification“. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, S. 1746–1751. DOI: 10.3115/v1/D14-1181.

[58] Diederik P. Kingma und Jimmy Lei Ba. „Adam: A Method for Stochastic Optimization“. In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. Hrsg. von Yoshua Bengio und Yann LeCun. 2015.

[59] James Kirkpatrick u. a. „Overcoming catastrophic forgetting in neural networks“. In: *Proceedings of the National Academy of Sciences of the United States of America* 114.13 (2017), S. 3521–3526. DOI: 10.1073/pnas.1611835114.

[60] Eric Knauss, Siv Houmb, Kurt Schneider, Shareeful Islam und Jan Jürjens. „Supporting Requirements Engineers in Recognising Security Issues“. In: *Requirements Engineering: Foundation for Software Quality: 17th International Working Conference, REFSQ*. Hrsg. von Daniel Berry und Xavier Franch. Springer Berlin Heidelberg, 2011, S. 4–18. DOI: 10.1007/978-3-642-19858-8_2.

[61] Eric Knauss und Daniel Ott. „(Semi-) automatic Categorization of Natural Language Requirements“. In: *Requirements Engineering: Foundation for Software Quality: 20th International Working Conference, REFSQ*. Hrsg. von Camille Salinesi und Inge van de Weerd. Springer International Publishing, 2014, S. 39–54. DOI: 10.1007/978-3-319-05843-6_4.

[62] Amy J. Ko, Thomas D. LaToza und Margaret M. Burnett. „A practical guide to controlled experiments of software engineering tools with human participants“. In: *Empirical Software Engineering* 20.1 (2015), S. 110–141. DOI: 10.1007/s10664-013-9279-3.

[63] Wei-Keat Kong, Jane Huffman Hayes, Alex Dekhtyar und Jeff Holden. „How Do We Trace Requirements: An Initial Study of Analyst Behavior in Trace Validation Tasks“. In: *Proceedings of the 4th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. Association for Computing Machinery, 2011, S. 32–39. DOI: 10.1145/1984642.1984648.

[64] Jennifer Krisch, Melanie Dick, Ronny Jauch und Ulrich Heid. „A Lexical Resource for the Identification of "Weak Words" in German Specification Documents". In: *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*. Hrsg. von Nicoletta Calzolari u. a. European Language Resources Association (ELRA), 2016.

[65] Jennifer Krisch und Frank Houdek. „The Myth of Bad Passive Voice and Weak Words: An Empirical Investigation in the Automotive Industry". In: *Proceedings of the 23rd IEEE International Requirements Engineering Conference (RE)*. IEEE, 2015, S. 344–351. DOI: 10.1109/RE.2015.7320451.

[66] Zijad Kurtanović und Walid Maalej. „Automatically Classifying Functional and Non-Functional Requirements Using Supervised Machine Learning". In: *Proceedings of the 25th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2017, S. 490–495. DOI: 10.1109/RE.2017.82.

[67] James Tin-Yau Kwok. „Automated Text Categorization Using Support Vector Machine". In: *Proceedings of the 5th International Conference on Neural Information Processing (ICONIP)*. Hrsg. von T. Omori und S. Usui. 1998, S. 347–351.

[68] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel und D. Henderson. „Handwritten Digit Recognition with a Back-Propagation Network". In: *Advances in Neural Information Processing Systems 2*. Hrsg. von David S. Touretzky. Morgan Kaufmann Publishers Inc., 1990, S. 396–404.

[69] Yann LeCun, Léon Bottou, Yoshua Bengio und Patrick Haffner. „Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86.11 (1998), S. 2278–2324. DOI: 10.1109/5.726791.

[70] Jiwei Li, Xinlei Chen, Eduard Hovy und Dan Jurafsky. „Visualizing and Understanding Neural Models in NLP". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. San Diego, California: Association for Computational Linguistics, 2016, S. 681–691. DOI: 10.18653/v1/N16-1082.

[71] Yonghong Li und Anil K. Jain. „Classification of Text Documents". In: *Proceedings of the 14th International Conference On Pattern Recognition (ICPR)*. Hrsg. von Anil K. Jain, Svetha Venkatesh und Brian C. Lovell. Bd. 2. 1998, S. 1295–1297. DOI: 10.1109/ICPR.1998.711938.

[72] Andrea De Lucia, Fausto Fasano, Rocco Oliveto und Genoveffa Tortora. „ADAMS Re-Trace: a Traceability Recovery Tool". In: *Proceedings of the 9th European Conference on Software Maintenance*

and Reengineering (CSMR). IEEE, 2005, S. 32–41. DOI: 10.1109/CSMR.2005.7.

[73] Laurens van der Maaten und Geoffrey Hinton. „Visualizing Data using t-SNE“. In: *Journal of Machine Learning Research* 9 (2008), S. 2579–2605.

[74] Christoph Maier, Alexander Schloske und Steffen Bothe. *Studie zur Funktionalen Sicherheit in der Automobilbranche (ISO 26262)*. Hrsg. von Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA. 2013. URL: https://www.ipa.fraunhofer.de/content/dam/ipa/de/documents/Publikationen/Studien/Studie_Funktionale_Sicherheit.pdf (besucht am 25. 09. 2020).

[75] Oded Maimon und Lior Rokach. „Introduction to Knowledge Discovery and Data Mining“. In: *Data Mining and Knowledge Discovery Handbook*. Hrsg. von Oded Maimon und Lior Rokach. Springer US, 2010, S. 1–15. DOI: 10.1007/978-0-387-09823-4_1.

[76] Raimundas Matulevičius. „How Requirements Specification Quality Depends on Tools: A Case Study“. In: *Advanced Information Systems Engineering: 16th International Conference, CAiSE*. Hrsg. von Anne Persson und Janis Stirna. Springer, 2004, S. 353–367. DOI: 10.1007/978-3-540-25975-6_26.

[77] Alistair Mavin, Philip Wilkinson, Sarah Gregory und Eero Uusitalo. „Listens Learned (8 Lessons Learned Applying EARS)“. In: *Proceedings of the 24th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2016, S. 276–282. DOI: 10.1109/RE.2016.38.

[78] Alistair Mavin, Philip Wilkinson, Adrian Harwood und Mark Novak. „EARS (Easy Approach to Requirements Syntax)“. In: *Proceedings of the 17th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2009, S. 317–322. DOI: 10.1109/RE.2009.9.

[79] Warren S. McCulloch und Walter Pitts. „A logical calculus of the ideas immanent in nervous activity“. In: *The bulletin of mathematical biophysics* 5.4 (1943), S. 115–133. DOI: 10.1007/BF02478259.

[80] Tomas Mikolov, Kai Chen, Greg Corrado und Jeffrey Dean. „Efficient Estimation of Word Representations in Vector Space“. In: *arXiv preprint* arXiv:1301.3781 (2013).

[81] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu und Jianfeng Gao. „Deep Learning Based Text Classification: A Comprehensive Review“. In: *arXiv preprint* arXiv:2004.03705 (2020).

[82] Tom M. Mitchell. *Machine Learning*. 1. Aufl. McGraw-Hill Education, 1997.

[83] Jakob Mund, Daniel Méndez Fernández, Henning Femmer und Jonas Eckhardt. „Does Quality of Requirements Specifications Matter? Combined Results of Two Empirical Studies“. In: *Proceedings of the 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. 2015, S. 144–153. DOI: 10.1109/ESEM.2015.7321195.

[84] Tsendsuren Munkhdalai und Hong Yu. „Neural Semantic Encoders“. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. Valencia, Spain, 2017, S. 397–407.

[85] Kamal Nigam, Andrew McCallum, Sebastian Thrun und Tom Mitchell. „Learning to Classify Text from Labeled and Unlabeled Documents“. In: *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI)*. USA: AAAI Press, 1998, S. 792–799.

[86] Thomas Noack, Thomas Karbe und Steffen Helke. „Reuse-Based Test Traceability: Automatic Linking of Test Cases and Requirements“. In: *International Journal on Advances in Software* 7.3&4 (2014), S. 469–485.

[87] Daniel Ott. „Automatic Requirement Categorization of Large Natural Language Specifications at Mercedes-Benz for Review Improvements“. In: *Requirements Engineering: Foundation for Software Quality: 19th International Working Conference, REFSQ*. Hrsg. von Joerg Doerr und Andreas L. Opdahl. Springer, 2013, S. 50–64. DOI: 10.1007/978-3-642-37422-7_4.

[88] Helmuth Partsch. *Requirements-Engineering systematisch*. 2. Aufl. Springer, 2010.

[89] Jeffrey Pennington, Richard Socher und Christopher D. Manning. „GloVe: Global Vectors for Word Representation“. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, S. 1532–1543. DOI: 10.3115/v1/D14-1162.

[90] Klaus Pohl. *Requirements Engineering: Grundlagen, Prinzipien, Techniken*. 2. Aufl. Heidelberg: dpunkt, 2008.

[91] Klaus Pohl und Chris Rupp. *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam Foundation Level / IREB compliant*. 2. Aufl. Rocky Nook, 2015.

[92] Project Smart. *The Standish Group Report - Chaos Report*. 2014. URL: https://www.projectsmart.co.uk/white-papers/chaos-report.pdf (besucht am 25. 09. 2020).

[93] J. R. Quinlan. „Induction of Decision Trees“. In: *Machine Learning* 1.1 (1986), S. 81–106. DOI: 10.1007/BF00116251.

[94] Marco Tulio Ribeiro, Sameer Singh und Carlos Guestrin. „Why Should I Trust You? Explaining the Predictions of Any Classifier“. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2016, S. 1135–1144. DOI: 10.1145/2939672.2939778.

[95] Anthony Robins. „Catastrophic Forgetting, Rehearsal and Pseudorehearsal“. In: *Connection Science* 7.2 (1995), S. 123–146. DOI: 10.1080/09540099550039318.

[96] Benedetta Rosadini, Alessio Ferrari, Gloria Gori, Alessandro Fantechi, Stefania Gnesi, Iacopo Trotta und Stefano Bacherini. „Using NLP to Detect Requirements Defects: An Industrial Experience in the Railway Domain“. In: *Requirements Engineering: Foundation for Software Quality: 23rd International Working Conference, REFSQ*. Hrsg. von Paul Grünbacher und Anna Perini. Springer, 2017, S. 344–360. DOI: 10.1007/978-3-319-54045-0_24.

[97] Frank Rosenblatt. „The perceptron: A probabilistic model for information storage and organization in the brain“. In: *Psychological Review* 65.6 (1958), S. 386–408. DOI: 10.1037/h0042519.

[98] David E. Rumelhart, Geoffrey E. Hinton und Ronald J. Williams. „Learning representations by back-propagating errors“. In: *Nature* 323 (1986), S. 533–536. DOI: 10.1038/323533a0.

[99] Bernhard Schölkopf und Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[100] Konstantinos Sechidis, Grigorios Tsoumakas und Ioannis Vlahavas. „On the Stratification of Multi-label Data“. In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD*. Hrsg. von Dimitrios Gunopulos, Thomas Hofmann, Donato Malerba und Michalis Vazirgiannis. Springer Berlin Heidelberg, 2011, S. 145–158. DOI: 10.1007/978-3-642-23808-6_10.

[101] Burr Settles. „Active Learning“. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6.1 (2012), S. 1–114. DOI: 10.2200/S00429ED1V01Y201207AIM018.

[102] Mohammad S. Sorower. *A Literature Survey on Algorithms for Multi-label Learning*. 2010.

[103] Nadya Stoyanova. „Verbesserung der Qualität natürlich-sprachlicher Spezifikationen“. Diss. Stuttgart: Universität Stuttgart, 2013.

[104] Daniel Strigl, Klaus Kofler und Stefan Podlipnig. „Performance and Scalability of GPU-Based Convolutional Neural Networks“. In: *Proceedings of the 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. Hrsg. von Marco Danelutto. IEEE, 2010, S. 317–324. DOI: 10.1109/PDP.2010.43.

[105] Lei Tang, Suju Rajan und Vijay K. Narayanan. „Large Scale Multi-Label Classification via MetaLabeler“. In: *Proceedings of the 18th International Conference on World Wide Web (WWW)*. Association for Computing Machinery, 2009, S. 211–220. DOI: 10.1145/1526709.1526738.

[106] The Standish Group International, Inc. *Chaos Report 2015*. 2015. URL: https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf (besucht am 25. 09. 2020).

[107] Sri Fatimah Tjong und Daniel M. Berry. „The Design of SREE — A Prototype Potential Ambiguity Finder for Requirements Specifications and Lessons Learned“. In: *Requirements Engineering: Foundation for Software Quality: 19th International Working Conference, REFSQ*. Hrsg. von Joerg Doerr und Andreas L. Opdahl. Springer, 2013, S. 80–95. DOI: 10.1007/978-3-642-37422-7_6.

[108] Fan-Yin Tzeng und Kwan-Liu Ma. „Opening the Black Box - Data Driven Visualization of Neural Networks“. In: *Proceedings of the 2005 IEEE Conference on Visualization (VIS)*. 2005, S. 383–390. DOI: 10.1109/VISUAL.2005.1532820.

[109] VDA Verband der Automobilindustrie e. V., Hrsg. *Automation: From Driver Assistance Systems to Automated Driving*. Berlin, 2015. URL: https://www.vda.de/dam/vda/publications/2015/automation.pdf (besucht am 25. 09. 2020).

[110] Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng und Paul J. Kennedy. „Training Deep Neural Networks on Imbalanced Data Sets“. In: *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN)*. 2016, S. 4368–4374. DOI: 10.1109/IJCNN.2016.7727770.

[111] Xin Wang und Ata Kabán. „Model-Based Estimation of Word Saliency in Text“. In: *Discovery Science: 9th International Conference, DS*. Hrsg. von Ljupčo Todorovski, Nada Lavrač und Klaus P. Jantke. Springer, 2006, S. 279–290. DOI: 10.1007/11893318_28.

[112] Jonas Paul Winkler. *Klassifikation von Anforderungen und Informationen zur Unterstützung von Qualitätssicherungsprozessen - Primärdaten empirische Studien*. 2020. DOI: 10.14279/depositonce-11105.

[113] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell und Anders Wesslén. *Experimentation in Software Engineering*. Springer Science & Business Media, 2012.

[114] Hui Yang, Anne de Roeck, Vincenzo Gervasi, Alistair Willis und Bashar Nuseibeh. „Analysing anaphoric ambiguity in natural language requirements“. In: *Requirements Engineering* 16.3 (2011), S. 163. DOI: 10.1007/s00766-011-0119-y.

[115] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola und Eduard Hovy. „Hierarchical Attention Networks for Document Classification“. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. San Diego, California: Association for Computational Linguistics, 2016, S. 1480–1489. DOI: 10.18653/v1/N16-1174.

[116] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs und Hod Lipson. „Understanding Neural Networks Through Deep Visualization“. In: *Deep Learning Workshop, 31st International Conference on Machine Learning*. 2015.

[117] Matthew D. Zeiler und Rob Fergus. „Visualizing and Understanding Convolutional Networks“. In: *Computer Vision – ECCV 2014: 13th European Conference*. Hrsg. von David Fleet, Tomas Pajdla, Bernt Schiele und Tinne Tuytelaars. Springer International Publishing, 2014, S. 818–833. DOI: 10.1007/978-3-319-10590-1_53.

[118] Matthew D. Zeiler, Graham W. Taylor und Rob Fergus. „Adaptive Deconvolutional Networks for Mid and High Level Feature Learning“. In: *Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV)*. 2011, S. 2018–2025. DOI: 10.1109/ICCV.2011.6126474.

[119] Ye Zhang und Byron C. Wallace. „A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification“. In: *arXiv preprint* arXiv:1510.03820 (2015).

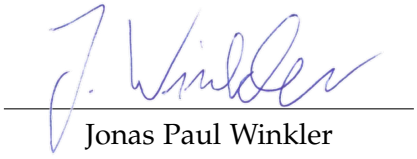
ERKLÄRUNG ZUR DISSERTATION

gemäß der Promotionsordnung vom 12. März 2020

Hiermit versichere ich an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel und Literatur angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Werken dem Wortlaut oder dem Sinn nach entnommen wurden, sind als solche kenntlich gemacht. Ich versichere an Eides statt, dass diese Dissertation noch keiner anderen Fakultät oder Universität zur Prüfung vorgelegen hat; dass sie - abgesehen von unten angegebenen Teilpublikationen und eingebundenen Artikeln und Manuskripten - noch nicht veröffentlicht worden ist sowie, dass ich eine Veröffentlichung der Dissertation vor Abschluss der Promotion nicht ohne Genehmigung des Promotionsausschusses vornehmen werde. Die Bestimmungen dieser Ordnung sind mir bekannt. Darüber hinaus erkläre ich hiermit, dass ich die Ordnung zur Sicherung guter wissenschaftlicher Praxis und zum Umgang mit wissenschaftlichem Fehlverhalten der Universität zu Köln gelesen und sie bei der Durchführung der Dissertation zugrundeliegenden Arbeiten und der schriftlich verfassten Dissertation beachtet habe und verpflichte mich hiermit, die dort genannten Vorgaben bei allen wissenschaftlichen Tätigkeiten zu beachten und umzusetzen. Ich versichere, dass die eingereichte elektronische Fassung der eingereichten Druckfassung vollständig entspricht.

Die Teilpublikationen sind auf Seite v vor dem Inhaltsverzeichnis der Dissertation aufgelistet.

Köln, 29. April 2021



Jonas Paul Winkler