INSTITUT FÜR INFORMATIK
Datenbanken und Informationssysteme

Universitätsstr. 1 D–40225 Düsseldorf

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Analyse von Online-Partizipationsverfahren: Textklassifikation auf mehreren Granularitätsebenen

Max Schubert

Masterarbeit

Beginn der Arbeit:	30. April 2018
Abgabe der Arbeit:	27. Dezember 2018
Gutachter:	Prof. Dr. Stefan Conrad
	Prof. Dr. Michael Schöttner

Erklärung

Hiermit versichere ich, dass ich diese Masterarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 27. Dezember 2018

Max Schubert

Zusammenfassung

Gegenstand dieser Arbeit ist die Analyse der Textklassifikation der Daten aus Online Partizipationsverfahren auf mehreren Granularitätsebenen. Online Partizipationsverfahren liefern häufig große Mengen an Vorschlägen und Kommentaren von Bürgern zu politischen Fragestellungen. Eine manuelle Auswertung dieser Daten während und nach dem Beteiligungszeitraum ist Zeit- und Arbeitsaufwändig. Dies motiviert die Verwendung von Methoden des Natural Language Processings zur Klassifikation von Textbeiträgen um semantisch interessante Inhalte wie Vorschläge, Argumente, Positionierungen oder Emotionen automatisiert zu extrahieren.

Diese Arbeit untersucht Klassifikationsaufgaben dieser Art auf zwei Datensätzen zu den Themen Braunkohleabbau im rheinischen Revier und der Nutzung des ehemaligen Flughafens Tempelhofer Feld in Berlin. Die Klassifikationsaufgaben werden dabei auf mehreren Granularitätsebenen, dass heißt mit verschiedenen Einheiten der Klassifikation, wie ganzen Beiträge, Sätzen oder einzelnen Wörter, vorgenommen und verglichen. Eine Auswahl an klassischen Machine Learning Verfahren mit verschiedenen Features sowie Deep Learning Algorithmen dienen als Klassifikatoren und werden hinsichtlich ihrer Fähigkeit, die Klassifikationsaufgaben durchzuführen, durch umfangreiche Experimente und manuelle inhaltliche Analyse beurteilt.

Dazu werden zuerst die Datensätze und der dahinterliegende Codierprozess beschrieben, verwandte Arbeiten vorgestellt und die in dieser Arbeit verwendeten Methoden erläutert. Darauf folgen Experimente und deren Auswertung auf drei Granularitätsebenen sowie Cross Plattform Untersuchungen. Es werden Verfahren und deren Parameter zur Realisierung der Klassifikation auf den verschiedenen Ebenen untersucht, Lösungen zu aufkommenden Problemen diskutiert sowie eigene Ideen zur Nutzung mehrerer Granularitätsebenen zugleich für eine Klassifikationsaufgabe untersucht.

Schließlich werden die gewonnen Erkenntnisse zusammengefasst und vor dem Hintergrund von zukünftigen Experimenten und den Intentionen der Betreiber von Online-Partizipationsplattformen ausgewertet und Vorschläge unterbreitet.

Im Anhang der Arbeit finden sich zudem detaillierte Ergebnisse zu allen Experimenten, welche aus Übersichtsgründen aus dem Evaluationskapitel verschoben wurden.

Insgesamt liefert die Arbeit Beurteilungen zu einzelnen Klassifikatoren und Features, schätzt die Schwierigkeit von verschiedenen Granularitätsebenen ein und liefert Vorschläge und Empfehlungen für Probleme, die im Zusammenhang mit der automatisierten Analyse von Online-Partizipationsplattformen auftreten können.

Inhaltsverzeichnis

1 Einleitung

In diesem Kapitel werden der Hintergrund der vorliegenden Arbeit motiviert, grundsätzliche Definitionen eingeführt und die Zielsetzung der Arbeit vorgestellt.

1.1 Motivation

Online-Diskussionen, wie sie in sozialen Netzwerken, Foren und insbesondere auf Online-Partizipationsplattformen zu finden sind, liefern große Mengen an Textdaten. Beiträge können dabei höchst unterschiedlich in Bezug auf ihren Inhalt ausfallen. Ob ein Beitrag argumentativ, themenrelevant, emotional oder respektlos ist, kann stark variieren und kann durch eine inhaltliche Analyse des Beitrags festgestellt werden. Ist eine zentrale Auswertung der Daten einer Online-Diskussion gewünscht, wie es insbesondere für Online-Partizipationsverfahren üblich ist, stellt die rein manuelle Auswertung der Daten einen erheblichen Arbeitsaufwand dar. In diesem Zusammenhang stellt sich die Frage, inwieweit eine solche Auswertung durch aktuelle Techniken des Natural Language Processing, insbesondere dem Teilbereich des Argument Minings, unterstützt werden kann. Auch die Datenaufbereitung für die Beteiligten der Diskussionen ist als Einsatzgebiet denkbar. So könnten Filter für argumentative Beiträge die Übersicht drastisch erhöhen und Filter bzw. Warnungen vor emotionalen Beiträgen und Respektlosigkeiten während der Moderation hilfreich sein.

Als Teil des NRW-Fortschrittskollegs Online-Partizipation[1] untersuchten Dr. Matthias Liebeck, Katharina Esau und Prof. Dr. Stefan Conrad bereits die Anwendung von Argument Mining für die Analyse in Online-Partizipationsverfahren. Dies geschah in Liebeck et al. (2016) und Liebeck (2018) auf einem reduzierten Korpus zum Online-Partizipationsverfahren Tempelhofer Feld (THF)[2]. Dabei wurden klassische Machine-Learning Algorithmen sowie neuronale Netze zur Klassifkation einzelner Sätze verwendet. Im Zuge dieser Arbeit sollen nun die von Dr. Matthias Liebeck durchgeführten Experimente auf einem größeren THF-Korpus sowie einem weiteren Korpus zum Braunkohleabbau[3], welche im Zuge der Doktorarbeit von Katharina Esau (siehe auch Esau (2018)) annotiert wurden, wiederholt werden. Zusätzlich soll die Übertragbarkeit der Ergebnisse zwischen zwei Plattformen im Zuge einer Cross-Platform-Evaluation ausgewertet werden. Außerdem erweitert diese Arbeit die vorangehenden Experimente um Klassfikationen auf weiteren Granularitätsebenen. Insbesondere die Aufgabe des Sequence-Taggings (Klassifikationen jedes einzelnen Wortes/Tokens) auf fein-granularer Ebene, sowie die Klassifikation ganzer Beiträge auf grob-granularer Ebene sollen hier durchgeführt werden. Schließlich werden eigene Ideen zur Verwendung einer fein-granularen Klassifikation für eine grob-granulare Klassifikationsaufgabe vorgestellt und direkt mit einer grob-granularen Klassifikation verglichen (zwischengranulare Vergleiche). Weitere Techniken wie Oversampling zur Balancierung der Klassen oder Gewichtung von Sequenzbestandteilen zur Verbesserung der Ergebnisse werden im Zuge der Arbeit ebenfalls behandelt.

[1] https://www.fortschrittskolleg.de/
[2] https://tempelhofer-feld-archiv.liqd.net/
[3] https://open.nrw/beteiligung-zur-leitentscheidung-braunkohle

1.2 Definitionen

Im Folgenden werden grundsätzliche Definitionen zum Themenbereich Argument Mining und Textklassifikation eingeführt.

1.2.1 Online-Partizipationsverfahren

Online-Partizipationsverfahren stellen Beteiligungsplattformen für politische Entscheidungen im Internet dar. Dabei werden textuelle und multimediale Vorschläge entweder direkt vorgegeben oder den Bürgern die Möglichkeit gegeben selbst Vorschläge zu einer Problemstellung zu formulieren. Realisiert wird dies ähnlich zu einem Online-Forum, sodass Vorschläge kommentiert und häufig auch bewertet werden können. Auch auf andere Kommentare kann wiederum Bezug genommen werden, sodass eine aktive Diskussion zwischen den Beteiligten üblich ist. Häufig ist zudem der Beteiligungszeitraum für ein Online-Partizipationsverfahren begrenzt und eine Auswertung durch die Initiatoren erfolgt üblicherweise nach dem Ablauf des Zeitrahmens. Beispiele für Online-Partizipationsverfahren sind das Beteiligungsverfahren zum Leitentscheid Braunkohle oder die Beteiligungsplattform zum Tempelhofer Feld (THF) in Berlin.

1.2.2 Argument Mining

Historisch bedingt stellt die Analyse von Argumentationsstrukturen bereits seit der Antike ein beliebtes Forschungsgebiet dar. Die Fragen zu beantworten welche Komponenten einer Diskussion zur Bildung einer Meinung führen, in welchen Zusammenstellungen diese Komponenten am wirksamsten sind um jemanden zu überzeugen und wie sich Texte verschiedensten Ursprungs auf semantisch sinnvolle Weise hinsichtlich dieser Qualitäten beschreiben lassen, bleiben schwierige Aufgaben. Der heutige Nutzen einer vollkommen maschinellen oder maschinell unterstützten Analyse von Argumentationen motiviert dabei vor allem vor dem Hintergrund von großen Textkorpora in Form von sozialen Netzwerken, Produktrezensionen, Verträgen und Gesetzeswerken, oder auch Online-Partizipationsverfahren die Entwicklung von performanten NLP-Verfahren.

Argument Mining stellt ein Anwendungsgebiet des Natural Language Processings (NLP) dar, welches sich mit der inhaltlichen Analyse von Textdaten bezüglich ihrer argumentativen Komponenten beschäftigt. Die automatisierte Erkennung von argumentativen Komponenten wie Vorschlägen, Argumenten, Positionierungen pro und contra zu einem Vorschlag sind dabei häufige Aufgaben. Auch wenn die konkrete Entwicklung von Argumentationsmodellen in den sozialwissenschaftlichen Arbeitsbereich fällt, stellt die Evaluation der Anwendbarkeit solcher Modelle in der Praxis Aufgaben, welche durch moderne NLP-Techniken bearbeitet werden können.

Ein bekanntes und viel diskutiertes (Hitchcock und Verheij, 2006) Argumentationsmodell des britischen Philosophen Stephen Toulmin (Toulmin, 2003) (siehe Abbildung 1) definiert grundlegend sechs Komponenten eines Argumentes zu einer gegebenen Problemstellung. „Data“ beschreibt dabei eine sachliche Darstellung der Sachlage. „Claim“ gibt die Überzeugung des Sprechers in Form eines Vorschlages wieder (z. B. „Bäume sollten nicht als Lärmschutz gepflanzt werden“). Um von den Randbedingungen, welche in

„Data“ stehen, nun eine schlüssige Argumentation zum „Claim“ zu ziehen, werden laut Toulmin häufig „Warrants“ verwendet, welche als Begründung des „Claims“ dienen. Ein solches „Warrant“ ist nicht zwingend erforderlich, stärkt aber die Überzeugungskraft des Arguments durch logische Deduktion. Außerdem können „Backing“ -Komponenten den eingesetzten „Warrant“ durch zusätzliche Informationen untermauern. Ferner kann ein „Qualifier“ zur Relativierung von Aussagen verwendet werden und eine „Rebuttal“ -Komponente nimmt mögliche Gegenargumente auf und entwertet diese durch eigene Argumente.

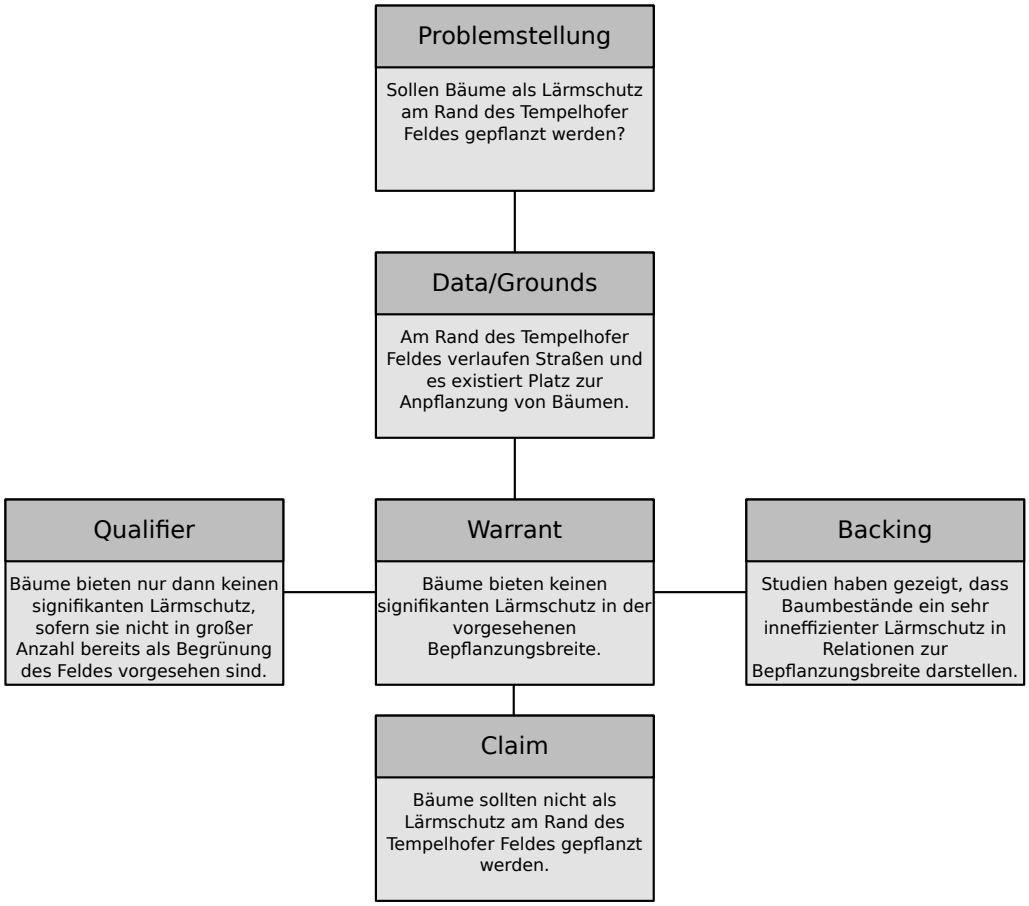


Abbildung 1: Beispiel für Toulmins Argumentationsmodell. Jeder Satz entspricht einer Argumentationskomponente. (Vereinfachte Version. Angelehnt an die Erläuterung zu Toulins Modell in *Toulmin Model of Argumentation* (2018))

In dieser Arbeit wird ein Argumentationsmodell von Liebeck et al. (2016) verwendet. Dieses Modell leitet sich von Freemans Claim-Premise Modell (Freeman, 2011) ab, berücksichtigt aber Erweiterungen des Modells, die insbesondere in Stab und Gurevych (2014) zur Untersuchung von argumentativen Essays entwickelt wurden. So verwenden Liebeck et al. (2016) ein dreiteiliges Modell, bestehend aus „Major Position“, „Premise“ sowie „Claim“ (siehe Abbildung 2).

„Major Positions“ stellen dabei Vorschläge dar, die von Nutzern oder den Initiatoren

	Positiv	Negativ
Argumentativ	Ich stimme dem Vorschlag zu.	Diese Idee halte ich für falsch.
Emotional	Das ist super, ich freue mich darauf!	Das ist eine Frechheit!

Tabelle 1: Beispiele für Sentiment Detection: Sätze, welche argumentativ bzw. emotional hinsichtlich des Sentiment als positiv bzw. negativ klassifiziert werden sollen

der Partizipationsplattform geäußert werden. Sie präsentieren meist eine Lösung zu einem bestehenden Problem in Form eines Vorschlags, von dem der Sprecher überzeugt ist (z. B. „Ich möchte einen ausgewiesenen Kitesurfbereich auf dem Tempelhofer Feld sehen“). „Claims“ repräsentieren Positionierungen von Nutzern zugunsten oder entgegen einer „Major Position“ (z. B. „Das ist eine gute Idee!“ oder „Dies halte ich für falsch.“). „Claims“ können auf diese Weise inhaltlich weiter in positive und negative „Claims“ unterschieden werden. Schließlich liefern „Premises“ Argumente, welche eine eigene „Major Position“ oder eigenen „Claim“ untermauern oder diejenigen anderer Diskussionsteilnehmer angreifen oder unterstützen können. Ein Beispiel für ein „Premise“ wäre „Ein ausgewiesener Kitesurfbereich würde die Offenheit des Feldes einschränken.“, was sich gegen das zuvor gegebene Beispiel einer „Major Position“ richten könnte.

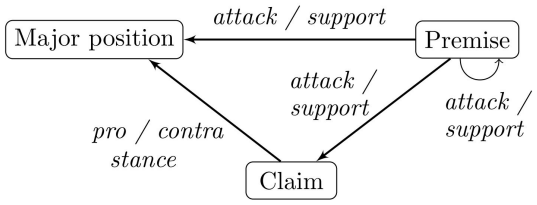


Abbildung 2: Argumentationsmodell, entnommen aus Liebeck et al. (2016)

1.2.3 Sentiment Detection

Sentiment Detection bezeichnet einen Teilbereich des Natural Language Processings bzw. Text Minings, welcher sich mit der Erkennung des Sentiments, d.h. der positiven oder negativen Haltung des Sprechers gegenüber einem Thema beschäftigt. Dabei kann sich die Sentiment Detection sowohl auf objektiv argumentative Merkmale konzentrieren oder aber auf emotionale Komponenten (siehe Tabelle 1).

1.2.4 Granularität des Taggings

Im Zuge dieser Arbeit werden mehrere Klassifikationsaufgaben auf zwei Korpora experimentell ausgewertet. Dabei unterscheiden sich die Klassfikationsaufgaben nicht nur durch verschiedene zu erkennende Klassen, sie werden zusätzlich auch auf verschiedenen Granularitätsebenenen (Tabelle 2) durchgeführt. In dieser Arbeit wird dabei zwischen drei Granularitätsebenen unterschieden:

Granularität	Fein	Mittel	Grob
Bezeichnung	Sequence-Tagging	Sentence-Tagging	Document-Tagging
Annotationsebene	Wort/Token	Satz	Dokument/Beitrag

Tabelle 2: Granularitätsebenen

In einer mittleren Granularitätsebene werden, wie bereits von Liebeck et al. (2017) durchgeführt, ganze Sätze mit einem Label klassifiziert und dieses Verfahren als Sentence-Tagging bezeichnet.

Auf grober Granularitätsebene werden hingegen ganze Dokumente (d.h. Kommentare im Kontext der Online-Partizipationsdatensätze) klassifiziert. Dieses Verfahren wird in dieser Arbeit fortan als Document-Tagging bezeichnet.

Schließlich erfolgt auch eine Klassifikation auf Token-/Wortebene was im Allgemeinen als Sequence-Tagging bekannt ist.

Die Granularität des Taggings beschreibt nun auf welcher Ebene das Klassifikationsproblem durchgeführt wird. Eine feine Granularität entspricht dem Sequence-Tagging, während eine grobe Granularität durch das Document-Tagging repräsentiert wird (siehe Tabelle 2). In dieser Arbeit werden zusätzliche Messungen über mehrere Granularitätsstufen vorgenommen (siehe Kapitel 1.2.4), sodass Ergebnisse aus einer feineren Granularitätsebene zur Bestimmung einer Klasse auf gröberer Granularitätsebene genutzt werden. Dieser Aspekt wird im folgenden als Intergranularität bezeichnet.

1.3 Zielsetzung und Aufbau der Arbeit

Ziel dieser Arbeit sind eine Evaluation der Übertragbarkeit der Ergebnisse von Liebeck (2018) auf zwei größere Korpora, die Einführung weiterer Klassfikationsaufgaben im Bereich der Sentiment Detection, ein experimenteller Vergleich gleicher Klassifikationsaufgaben auf verschiedenen Granularitätsebenen sowie Experimente zur Nutzung von feingranularen Ergebnissen zur Klassfikation auf grob granularer Ebene. Inhaltlich ist die Arbeit dazu wie folgt aufgebaut:

Kapitel 2 beschreibt die Erzeugung und Eigenschaften der verwendeten Datensätze. Dazu wird die vorangehende Zusammenarbeit mit Katharina Esau, Dr. Matthias Liebeck, Sarah-Michelle Nienhaus und Tanja Tix zur Entwicklung eines Codebuches, Durchführung eines Codierprozesses und Vorverarbeitung des Datensatzes erläutert sowie statistische Informationen zum Datensatz geliefert.

In Kapitel 3 werden verwandte Arbeiten im Bereich des Argument Mining behandelt. Gemeinsamkeiten bezüglich der Anwendungsdomäne von Online-Partizipationsverfahren als auch bezüglich der betrachteten Granularität werden hierbei herausgestellt.

Kapitel 4 beschreibt das verwendete Framework, welches von Liebeck (2018) entwickelt wurde und im Zuge dieser Arbeit um die Kompatibilität mit den neuen Datensätzen sowie Verwendung von verschiedenen Granularitätsebenen erweitert wurde. Hierbei werden einige grundlegende Verfahren aus dem Bereich des klassischen Machine Learning sowie Deep Learning beschrieben, die Realisierung verschiedener Granularitätsebenen

präsentiert sowie auf die Verwendung von Oversampling (d.h. der künstlichen Erweiterung der Daten zur Balancierung der Klassen) eingegangen.

In Kapitel 5 folgt die Auswertung der Ergebnisse für jede Klassfikationsaufgabe, Granularitätsebene, die Untersuchung der intergranularen Messungen sowie für den Einfluss von Oversampling zur Klassenbalancierung.

Schließlich folgt in Kapitel 6 eine Zusammenfassung und ein abschließendes Fazit mit Ausblick auf mögliche zukünftige Arbeiten.

2 Datensätze

Im Folgenden werden die in der Arbeit verwendeten Datensätze sowie das zugrundeliegende Annotationsschema und Argumentationsmodell beschrieben. Beide Datensätze sind im Zuge der noch nicht veröffentlichten Doktorarbeit von Katharina Esau (siehe auch (Esau, 2018)) und im Rahmen des NRW Fortschrittskollegs Online-Partizipation entstanden.

2.1 Online-Partizipationsverfahren

Dieser Arbeit liegen zwei Online-Partizipationsverfahren zu Grunde. Das erste Verfahren wurde bereits auszugsweise von Liebeck et al. (2016) untersucht und beschäftigt sich mit der Nutzung des ehemaligen Flughafengebietes Tempelhofer Feld (THF) in Berlin. Das zweite Verfahren stellt Änderungen zu den Leitentscheidungen der Braunkohlenutzung im Bereich Garzweiler zur Diskussion und wird hier als zweiter Korpus hinzugefügt.

2.1.1 Tempelhofer Feld

Unter dem Namen Tempelhofer Feld wird das seit 2008 stillgelegte Gelände des ehemaligen Flughafens Berlin-Tempelhof verstanden. Dieses über 300 Hektar große Gelände wurde in der Vergangenheit als Ackerfläche, militärischer Übungsplatz und schließlich als Flughafengelände genutzt. Nach der Öffnung für die Öffentlichkeit im Jahr 2010 wurde das Gelände als Park- und Freizeitgelände genutzt und bietet dank großer, offener Flächen Platz für verschiedenste Freizeitaktivitäten.

Seit der Öffnung entwickelten sich Pläne der Stadt Berlin zur Bebauung und kommerziellen Nutzung des Geländes, was auf Ablehnung unter der Berliner Bevölkerung stieß. Die im Jahr 2011 gegründete Bürgerinitiative „100% Tempelhofer Feld“ erzielte mit einem Volksentscheid aus dem Jahr 2014 Erfolg. Mit dem Gesetz zum Erhalt des Tempelhofer Feldes wurde eine Bebauung des Geländes untersagt und die Nutzung der Fläche als Erholungsgebiet festgesetzt. Zwischen 2014 und 2017 wurde daraufhin eine Online-Plattform[4] zur Sammlung und Aufbereitung von Vorschlägen und Diskussionen über den Erhalt, Pflege und Nutzung des Tempelhofer Feldes erstellt. Dort entstanden Vorschläge und Diskussionen zu den Kategorien „Bewirtschaftung“, „Erinnerung“, „Freizeit“, „I ♡ THF“, „Mitmachen“, „Natur“ und „Was vergessen?“. Die Beiträge zu diesen Kategorien liefern Textdaten in Form eines Initialvorschlages und ggf. dazu geäußerten Kommentaren sowie Meta-Informationen, wie die angegebene Meinung des Nutzers zum Thema („Dafür“ oder „Dagegen“), den angegebenen Nutzernamen, Datum und Bezugskommentar, falls es sich um eine Antwort auf einen Kommentar oder auf den Initialvorschlag handelt. Die Kommentare sind dabei in einer Baumstruktur angeordnet,

[4] https://tempelhofer-feld-archiv.liqd.net/

Kategorie	Vorschläge	Kommentare
Was vergessen?	26	64
I ♡ THF	22	71
Nicht eingeordnet	7	15
Bewirtschaftung	65	289
Mitmachen	45	140
Freizeit	103	767
Natur	33	185
Erinnerung	19	97
$\sum$	320	1308

Tabelle 3: Datensatz Tempelhofer Feld

sodass ein Nutzer direkt auf einen anderen Beitrag antworten kann. Grundlegende Kennzahlen zum Datensatz finden sich in Tabelle 3. Dabei ist zu beachten, dass einige Daten im Zuge des Codierprozesses ausgefiltert wurden, sodass sich die Angaben in Tabelle 3 auf die von Katharina Esau erhobenen und auch in dieser Arbeit verwendeten Daten beziehen. Insgesamt liefert der THF-Korpus somit 1628 Beiträge und 122996 Token.

2.1.2 Braunkohle-Dialog

Das Gebiet um den ehemaligen Ort Garzweiler, etwa 12km südlich von Mönchengladbach, wird seit dem 19. Jahrhundert als Braunkohleabbaugebiet genutzt. In zwei Leitentscheidungen in den Jahren 1987 und 1991 wurden dabei die Abbaugrenzen des Tagebaus vergrößert und im Zuge dessen der Abriss und die Umsiedlung vieler im Gebiet liegender Ortschaften beschlossen.

Aufgrund der energiepolitischen Wende hin zu erneuerbaren Energien entstanden 2014 Pläne der Landesregierung die Abbaugrenzen des Abbaugebietes Garzweiler II zu verkleinern, sodass die geplante Umsiedlung der Ortschaften Holzweiler und Dackweiler hinfällig wurden. Da die Ankündigung dieser Pläne bereits im Vorfeld zu Diskussionen unter den Betroffenen führte, wurde 2015 eine Online-Plattform für die Beteiligung zur Leitentscheidung Braunkohle[5] eingerichtet, auf der zwischen dem 30.09.2015 und 08.12.2015 Interessierte ihre Meinung zu den vier Entscheidungssätzen diskutieren konnten. Eine zentrale Auswertung der Daten durch die Landesregierung NRW folgte anschließend im Jahr 2016, bevor im Juli 2016 schließlich die Details der Verkleinerung des Abbaugebietes durch die neue Leitentscheidung beschlossen wurden.

Inhalte der Diskussion stellten dabei die allgemeine Verkleinerung des Abbaugebietes in Entscheidungssatz 1, die zukünftige Nutzung des Tagebausgebietes als Restsee in Entscheidungssatz 2, detaillierte Pläne zur Lösung der infrastrukturellen Probleme in den Sätzen 3.1 bis 3.4 sowie zukünftige Entwicklungen und Pläne im rheinischen Revier in Entscheidungssatz 4 dar.

Grundlegende Daten zum Datensatz finden sich in Tabelle 4. Insgesamt enthält der co-

[5] https://open.nrw/beteiligung-zur-leitentscheidung-braunkohle

Entscheidungssatz	Kommentare
1 (Verkleinerung des Abbaugebietes)	902
2 (Nutzung des Gebiets als Restsee)	93
3.1 (Mindestabstand zur Abbaugrenze)	155
3.2 (Anbindung der Orte durch Straße L19)	30
3.3 (Ufergestaltung des Restsees)	9
3.4 (Existenz der Landwirtschaftsbetriebe)	21
$\sum$	1210

Tabelle 4: Datensatz Braunkohle

dierte Braunkohle-Korpus 1216 Beiträge und 188813 Token. Auch hier gilt, dass es sich um die von Katharina Esau erhobenen Daten handelt und nicht um alle verfügbaren Beiträge. So wurde z.B. Entscheidungssatz 4 sowie einzelne Vorschläge aus den hier verwendeten Korpora herausgefiltert, da diese für die weiteren Analysen von Katharina Esau nicht relevant waren. Gründe dafür waren eine geringe Anzahl an Beiträgen, themenfremde Diskussionen (z.B. Metadiskussionen über die Beiteilungsplattform) oder das Fehlen einer strukurierten Diskussion (z.B. nur Zustimmungen durch ein Wort).

2.2 Codebuch

Im Zuge ihrer Doktorarbeit (siehe auch Esau (2018)) entwickelte Katharina Esau ein Codebuch, auf dessen Grundlage auch alle in dieser Arbeit vorgenommenen Klassifikationsaufgaben basieren. Grundlage der Analyse von Argumentationstrukturen stellt dabei im sozialwisschenschaftlichen Bereich ein Codebuch dar, welches die zu annotierenden Klassen definiert und Hilfestellungen zur Annotation liefert. Die Entwicklung des hier verwendeteten Codebuches durch Katharina Esau verlief im Zuge der Projektarbeit des Autors in ständiger Zusammenarbeit mit Matthias Liebeck, Sarah-Michelle Nienhaus und Tanja Tix. Dabei wurden die grundlegenden Definitionen der Klassen erarbeitet, über die Möglichkeit und das Verbot von Klassenüberschneidungen diskutiert, Randfälle und Beispiele ausgearbeitet und insgesamt ein gemeinsames Verständnis der Codiergrundlage erzeugt. Die gemeinsame Entwicklung des Codebuches führte auf diese Weise nicht nur zur Verfügbarkeit eines hilfreichen Nachschlagewerks während der Codierung, sondern auch zu einem instinktiven Verständnis der zugrundeliegenden Semantik der einzelnen Klassen, sodass ein gutes Inter-Annotator-Agreement schnell erreicht werden konnte.

2.2.1 Variablen

Die in dieser Arbeit verwendeten Klassen, welche im sozialwissenschaftlichen Umfeld als Variablen bezeichnet werden, wurden von Katharina Esau im Codebuch dokumentiert und stellen die Grundlage des Codierprozesses dar, welcher 2017 erfolgte. Als Variablen werden dabei alle semantisch interessanten Informationen zu Beiträgen bezeichnet, sodass nicht nur die während der Annotationsphase verwendeten Klassenlabel, sondern auch Zusatzinformationen, wie Zeitstempel und Nutzername, als Variablen bezeichnet

werden. Eine Übersicht über die verwendeten Variablen findet sich in Tabelle 5. Die grau markierten Variablen wurden dabei nicht manuell codiert, sondern direkt dem Datensatz entnommen.

Die Variablen werden durchnummeriert (Variablennummer), mit einem eindeutigen Kürzel (Variablenkürzel) versehen und können gegebenenfalls eine der in der Spalte „Code“ aus Tabelle 5 angegebenen Ausprägungen annehmen. Zusätzlich wurde für jede Variable die Ebene der Codierung unterschieden. Variablen 12-15 auf Beitragsebene wurden dabei durch das Hinzufügen von zusätzlichen Zeilen zu jedem Beitrag realisiert. Diese Zusatzzeilen konnten dann als komplette Zeile annotiert werden und mit einem der zugehörigen Codes als Zusatzinformation versehen werden. Variablen auf Aussageebene entsprechen den Klassen eines Sequence-Tagging-Prozesses. Jeder Teilbereich eines Beitrages konnte auf Zeichenebene mit den jeweiligen Klassen annotiert werden. Für diejenigen Variablen, welche auf Aussage- und Relationsebene codiert wurden, konnten analog jeder Teilbereich eines Beitrages auf Zeichenebene annotiert werden, wobei hier zusätzliche Informationen zu Relationen zwischen annotierten Bereichen auch über mehrere Beiträge hinweg angegeben werden konnten. Genaue Informationen zu den Variablen und ihrer Bedeutung finden sich in der Doktorarbeit von Katharina Esau (siehe auch Esau (2018)). Erste inhaltliche Analysen finden sich bereits in Esau (2018).

Variablennummer	Variablenkürzel	Variablenname	Code	Ebene der Codierung
1	CNR	Codierernummer	Code: 1 = Katharina Esau; 2 = Matthias Liebeck; 3 = Sarah-Michelle Nienhaus; 4 = Max Schubert; 5 = Tanja Tix	Beitrag
2	BNR	Beitragsnummer	<4-stellige Beitragsnummer>	Beitrag
3	BART	Beitragsart	Code: 0 = Initialbeitrag; 1 = Kommentar	Beitrag
4	WORT	Wörterzahl	<Anzahl der Wörter>	Beitrag
5	ZEIT	Zeitstempel	TT/MM/JJJJ HH:mm:ss	Beitrag
6	ÜBTHEM	Übergeordnetes Thema	Code: 1 = Tempelhofer Feld; 2 = Braunkohle	Beitrag

7	UNTHEM	Unterthema	Code: 1 = Bewirtschaftung; 2 = Erinnerung; 3 = Freizeit; 4 = I ♡ THF; 5 = Mitmachen; 6 = Natur; 7 = Was vergessen?; 8 = Entscheidungssatz 1; 9 = Entscheidungssatz 2; 10 = Entscheidungssatz 3.1; 11 = Entscheidungssatz 3.2; 12 = Entscheidungssatz 3.3; 13 = Entscheidungssatz 3.4; 14 = Entscheidungssatz 4	Beitrag
8	PROST	Pro Stimmen	<Anzahl Pro Stimmen>	Beitrag
9	CONST	Contra Stimmen	<Anzahl Contra Stimmen>	Beitrag
10	BEW	Bewertung	<Anzahl Pro Stimmen - Anzahl Contra Stimmen>	Beitrag
11	SPRTYP	Sprechertyp	Code: 0 = Moderator; 1 = Nutzer	Beitrag
12	THEMB	Themenbezug	Code: 0 = Kein Themenbezug; 1 = Themenbezug vorhanden	Beitrag
13	IDENT	Identität	Code: 0 = Pseudonym; 1 = Klarname; 99 = gelöscht	Beitrag
14	GSCH	Geschlecht	Code: 0 = Weiblich; 1 = Männlich; 99 = Unklar/ Nicht vorhanden	Beitrag
15	GESAMTH	Gesamthaltung	Code: 0 = Dafür; 1 = Dagegen; 2 = Neutral; 99 = Unklar	Beitrag
16	VORSCH	Vorschlag		Aussage
17	POSPR	Positionierung Pro		Aussage
18	POSCO	Positionierung Contra		Aussage
19	ARG	Argument		Aussage
20	NAR	Narration		Aussage
21	EMOP	Emotion Positiv		Aussage

22	EMON	Emotion Negativ		Aussage
23	HUM	Humor		Aussage
24	GREET	Greeting		Aussage
25	RES	Respektlosigkeit		Aussage
26	INFFRA	Informationsfrage		Aussage
27	BEGFRA	Begründungsfrage		Aussage
28	LÖS	Lösungsvorschlag		Aussage
29	REFL	Reflexivität		Aussage + Relation
30	EMP	Empathie		Aussage + Relation
31	BEZ	Bezugnahme		Aussage + Relation

Tabelle 5: Variablen nach Katharina Esau. Die Variablen 1-11 (grau hinterlegt) wurden nicht manuell codiert, sondern als Metadaten dem Datensatz entnommen.

Im Speziellen sind für diese Arbeit die Variablen „Vorschlag“, „Argument“, „Positionierung Pro“, „Positionierung Contra“ sowie „Emotion Positiv“ und „Emotion Negativ“ relevant und werden für die Klassifikationsaufgaben verwendet. Die Variable Vorschlag entspricht dabei der in Liebeck et al. (2016) beschriebenen „Major Position“, die beiden Variablen Positionierung Pro und Positionierung Contra entsprechen den Definitionen von positiven bzw. negativen „Claims“ und die Variable Argument der Klasse „Premise“.

2.3 Codierprozess

Der Annotationsprozess, auch Codierprozess genannt, wurde in den Monaten August und September 2017 von Katharina Esau, Matthias Liebeck, Sarah-Michelle Nienhaus, Tanja Tix und im Zuge der Projektarbeit des Autors der Arbeit durchgeführt (Esau, 2018).

Zur Codierung selbst wurde dabei das Annotationstool Brat[6] von Stenetorp et al. (2012) verwendet, welches eine Weboberfläche zur zeichenbasierten Annotation von Texten liefert. Zur effektiven Nutzung dieses Tools waren einige Vorverarbeitungsschritte nötig. Einerseits mussten für alle Beiträge in den als JSON-Dateien vorliegenden Korpora einzelne Text- sowie Annotationsdateien erzeugt werden. Dazu mussten die Beiträge in Sätze gespalten werden, was durch spaCy[7] sowie eine manuelle Kontrolle jedes Satzes erfolgte. Da zudem die Erreichbarkeit der Online-Plattformen zu den Beteiligungsverfahren während des Codierprozesses nachließ, war eine zusätzliche Visualisierung der Korpora durch HTML-Darstellungen nötig, um den Annotatoren die Möglichkeit zu geben, die Diskussion so zu lesen, wie sie auf der ursprünglichen Plattform repräsentiert wurde. Ebenfalls wichtig war die Entwicklung einer eigenen Syntax zur Codierung von Relationen. So konnten Variablen, wie z.B. „Bezugnahme“, zusätzliche Informationen

[6] http://brat.nlplab.org/
[7] https://spacy.io/

beinhalten, wie die ID des Kommentars, auf die Bezug genommen wird, und die dort im speziellen referierten Variablen, was über das Annotationstool Brat ohne diese Syntax nicht möglich gewesen wäre. Ein Beispiel dafür wäre die Codierung eines Satzes mit dem Label „Bezugnahme“und Eintrag in das Kommentarfeld „c899 NAR, ARG“mit der Bedeutung, dass es sich um eine Bezugnahme auf Kommentar mit ID c899 und den als Narration und Argument codierten Inhalt handelt.

Zur Überprüfung des Inter-Annotator Agreements wurden Auszüge aus den Entscheidungssätzen des Braunkohlekorpus sowie einzelne Vorschläge aus dem THF-Korpus ausgewählt, um in einer Probecodierung von allen Annotatoren codiert zu werden. Nach Überprüfung der Übereinstimmung und weiterer Schulung der Annotatoren wurde eine weitere Probecodierung durchgeführt, welche den Erwartungen von Katharina Esau an den ermittelten Agreement-Score bezüglich Krippendorffs Alpha (Krippendorff, 2004, Krippendorff, 2011) von 0.75 für Argumente erfüllte und somit die Nutzbarkeit der später codierten Daten hinsichtlich des Agreements verifizierte. Informationen zum Agreement finden sich in den Tabellen 6, 7 und 8 sowie Tabelle 9, welche das Agreement auf Zeichenebene, analog zur Verfahrensweise in Habernal und Gurevych (2017), mit Hilfe von Krippendorffs unitized α_u berechnet. Für Tabelle 9 wird dabei Krippendorffs unitized α_u verwendet, welches die Grenzen der Annotationen auf Zeichenebene berücksichtigt und alle Dokumente konkateniert betrachtet. Kategorie HAUPT beinhaltet die Klassen Vorschlag, Argument, Positionierung Pro und Contra, Kategorie EMO beinhaltet die Klassen Emotion Positiv und Negativ.

Hinsichtlich der Agreement-Scores für die einzelnen Variablen fällt auf, dass vor allem Positionierung Contra schlechter erkannt wurde als die anderen Variablen, während Vorschläge durchweg gut erkannt wurden. Dies kann einerseits daran liegen, dass negative Positionierungen häufig auch als Argument gegen einen Vorschlag vorgetragen werden und somit eher als Argument annotiert werden (da die Variablen Vorschlag, Argument sowie Positionierungen nicht gleichzeitig annotiert werden dürfen, muss sich der Annotator hier entscheiden). Andererseits kann dies gerade im Falle der Braunkohlediskussion auch an der vorliegenden Grundstimmung liegen, die selbst bei Zustimmung negative Äußerungen zulässt. Vorschläge wurden hingegen gut erkannt, was gerade im Fall des THF-Datensatzes der klaren Aufteilung in Initialbeiträge und Kommentare geschuldet sein könnte, während im Braunkohlekorpus nur wenige Vorschläge außerhalb der Leitentscheidungen vorgetragen werden.

Nach den Probecodierungen erfolgte dann die Hauptcodierung der von Katharina Esau ausgewählten Vorschläge bzw. Entscheidungssätze der beiden Datensätze. Dadurch entstanden 1308 annotierte Kommentare aus dem Korpus Tempelhofer Feld sowie 1210 annotierte Kommentare aus dem Korpus der Leitentscheidung Braunkohle.

Die auf diese Weise erzeugten Daten wurden anschließend in mehreren Iterationen auf syntaktische Fehler überprüft, wofür ein Tool zur Erkennung und Visualisierung von Fehlern implementiert wurde. Schließlich wurden die Daten ausgelesen und für Katharina Esau verarbeitet und in verschiedenen Formaten für die weitere Untersuchung unter sozialwissenschaftlichen Gesichtspunkten exportiert.

Variable	Fleiss κ	Krippendorff α	Prozentuales Agreement	Vorkommen	Vorkommen prozentual
Vorschlag	0.82	0.82	0.92	209	29,86 %
Argument	0.75	0.75	0.90	495	70,71 %
PositionierungPro	0.73	0.73	0.93	110	15,71 %
PositionierungContra	0.58	0.58	0.88	117	16,71 %
EmotionPositiv	0.73	0.73	0.93	105	15,00 %
EmotionNegativ	0.78	0.78	0.92	165	23,57 %

Tabelle 6: Inter-Annotator Agreement auf Dokumentebene für die in dieser Arbeit relevanten Variablen.

Variablenkategorie	Fleiss κ	Krippendorff α	Prozentual
HAUPT	0.64	0.64	0.78
EMO	0.65	0.65	0.94

Tabelle 7: Inter-Annotator Agreement auf Tokenebene für die in dieser Arbeit relevanten Variablen. HAUPT ist die Vereinigung aus Vorschlag, Argument und positiven sowie negativen Positionierungen. EMO die Vereinigung aus positiven und negativen Emotionen.

Variable	Vorkommen in HAUPT	Vorkommen in EMO
Vorschlag	5410	0
Argument	24514	0
Positionierung Pro	1119	0
Positionierung Contra	1617	0
Emotion Positiv	0	943
Emotion Negativ	0	3067
Keine Variable aus Gruppe	10830	39480

Tabelle 8: Verteilung der Klassen auf Tokenebene in den Variablenkategorien HAUPT und EMO in der Probecodierung.

Variable	Krippendorff α unitized
HAUPT	0.62
Vorschlag	0.34
Argument	0.66
PositionierungPro	0.50
PositionierungContra	0.53
EMO	0.63
EmotionPositiv	0.39
EmotionNegativ	0.66

Tabelle 9: Inter-Annotator-Agreement auf Span-Ebene für die in dieser Arbeit relevanten Variablen.

Klasse	THF	THF %	Braunkohle	Braunkohle %
Argumentativ (VORSCH, ARG, POSPRO, POSCO)	1445	88,76 %	1027	84,11 %
Positionierungen (POSPRO, POSCO)	532	32,68 %	209	17,12 %
Emotional (EMOP, EMON)	459	28,19 %	377	30,88 %
Vorschlag	633	38,88 %	99	8,11 %
Argument	1033	63,45 %	732	59,95 %
Positionierung Pro	397	24,39 %	85	6,96 %
Positionierung Contra	150	9,21 %	146	11,96 %
Emotion Positiv	308	18,92 %	113	10,08 %
Emotion Negativ	190	11,67 %	299	24,49 %
Alle Beiträge	1628	100 %	1221	100 %

Tabelle 10: Statistik zu den codierten Datensätzen. Einträge geben die Anzahl der Beiträge an, in denen die entsprechende Klasse codiert wurde bzw. den Anteil an allen Beiträgen.

2.4 Statistik

Eine Übersicht über die durch die Codierung erhaltenen Klassenverteilungen auf Satzebene ist in Tabelle 10 zu finden. Tabelle 11 liefert weitere Informationen über die später verwendeten Trainings- und Testsets.

Auffällig im Vergleich zwischen den beiden Korpora ist hierbei das Verhältnis von positiven zu negativen Emotionen. Der Braunkohlekorpus weißt deutlich mehr negative Emotionen auf, was sich aus zu einem großen Teil aus der persönlichen Betroffenheit der Anwohner und den vorgegebenen Vorschlägen (Leitentscheidungen) ableiten lässt. Im Gegensatz zum THF-Korpus, wurden hier keine expliziten Möglichkeiten zur Erstellung von Vorschlägen gegeben, sodass viele Bürger ihre Unzufriedenheit mit dem jeweiligen Sachverhalt nur durch eigene Kommentare ausdrücken konnten und keine expliziten Vorschläge vorfanden, die ihre Meinung widerspiegelten. Der starke Unterschied in codierten Vorschlägen würde dies bestätigen, berücksichtigt dabei allerdings nicht, dass die THF-Plattform das Erstellen von Vorschlägen ermöglichte. Werden nur diejenigen Vorschläge im THF-Korpus gezählt, die nicht in einem Initialbeitrag vorkommen, ergeben sich 331 Vorschläge, was immer noch einem Anteil von 20,33 % entspricht und damit deutlich höher liegt, als die 8,11 % der Beiträge im Braunkohle Korpus, die Vorschläge enthalten. Inwieweit dies die Ergebnisse der Cross-Platform Evaluationen beinflusst, ist in Kapitel 5 ausgeführt. Ebenfalls auffällig sind die deutlich längeren mittleren Satzlängen im Braunkohlekorpus. Dies deckt sich mit den Erfahrungen der Codierung: Beiträge im Braunkohlekorpus teilen meist (teils auch aufgrund der geringen Anzahl an verfügbaren Vorschlägen) ihren Inhalt in längerer Form mit einem größeren Anteil an Erklärungen zum Kontext dar, während Beiträge im Korpus THF häufig kurze Angaben zur eigenen Positionierung darstellen. Der Einfluss dieser Eigenschaften wird ebenfalls in Kapitel 5 behandelt.

Datensatz	Kommentare	∅ Beitragsl.	Max. Beitragsl.	Min. Beitragsl.	∅ Satzl.	Max. Satzl.	Min. Satzl.
THF Test	1112	6.7	69	1	76.7	865	1
THF Train	517	7.1	138	1	82.2	1669	2
BRK Test	827	7.7	58	1	155.2	1100	2
BRK Train	396	8.0	48	1	155.8	953	2
Both Test	1938	7.2	69	1	110.4	1100	1
Both Train	913	7.5	138	1	114.0	1669	2

Tabelle 11: Statistik zu den codierten Datensätzen. Angegebene Beitragslängen werden in der Anzahl von Sätzen gemessen, Satzlängen in der Anzahl an Tokens.

Subtask	Klassen	Einschränkung
A	Argumentative Non-argumentative	Keine
B	Major Position Premise Claim	Argumentativ
C	Claim Pro Claim Contra	Claim
D	Emotional Non-emotional	Keine
E	Emotion positiv Emotion negativ	Emotional

Tabelle 12: Übersicht über die Klassifikationsaufgaben in dieser Arbeit. Die Spalte Einschränkung gibt an, auf welche Klassen sich die jeweilige Klassifikationsaufgabe beschränkt. Die farbliche Hinterlegung entspricht den entsprechenden Farben der Subtasks in den Plots in Kapitel 5

2.5 Klassifikationsaufgaben

Die in dieser Arbeit untersuchten Klassifikationsaufgaben entsprechen weitestgehend denjenigen aus Liebeck (2018). Dabei wurden drei Aufgaben, als Subtasks bezeichnet, untersucht. Subtask A ist eine binäre Klassfikationsaufgabe zwischen den Klassen „argumentativ“ und „nicht argumentativ“, wobei als argumentativ die Variablen „Vorschlag“, „Argument“, „Emotion positiv“ und „Emotion negativ“ angesehen werden.

Subtask B operiert auf einem auf argumentative Inhalte eingeschränkten Korpus und unterscheidet zwischen „Major Position“, „Claim“ und „Premise“ mit den entsprechenden Variablen (siehe Kapitel 2.2.1). Die Einschränkung auf argumentative Inhalte ist dabei durch das in Liebeck (2018) vorgeschlagene mehrschichtige Vorgehen während der Klassifikation begründet. Ähnliche Einschränkungen der Korpora werden auch für die restlichen Subtask verwendet. Ausnahmen dafür werden in einzig in Kapitel 5.2 vorgenommen.

Subtask C unterscheidet dann auf einem auf „Claims“ eingeschränkten Korpus, zwischen „Claim Pro“ und „Claim Contra“.

Zusätzlich werden in dieser Arbeit die Subtasks D und E eingeführt. Subtask D ist eine binäre Klassfikationsaufgabe zwischen emotionalen und nicht emotionalen Textstellen, während Subtask E auf den emotionalen Textstellen zwischen positiven und negativen Emotionen unterscheiden soll. Eine Übersicht über die Klassifikationsaufgaben findet sich in Tabelle 12.

Zudem werden alle Klassifikationsaufgaben auf beiden Korpora und der Vereinigung beider Korpora durchgeführt. Zusätzlich werden im Sinne einer Cross-Platform Evaluation Klassifikatoren, die auf einem Korpus trainiert wurden, auf dem jeweils anderen Korpus getestet, um die Übertragbarkeit des Trainings zu beurteilen.

Alle Klassifikationsaufgaben werden in dieser Arbeit auf drei Granularitätsebenen (siehe

Kapitel 1.2.4) durchgeführt. Analog zu Liebeck et al. (2016) und Liebeck (2018), werden auf mittlerer Granularität ganze Sätze klassifiziert. Zusätzlich wird nun auch feingranulares Sequence Tagging und grob granulares Document Tagging eingesetzt.

Die Anwendbarkeit von intergranularem Tagging wird ebenfalls untersucht. Dazu werden die Ergebnisse einer feiner granularen Klassifikation verwendet, um die Klassen einer gröber granularen Klassifikationsaufgabe zu bestimmen. Die damit erzielten Ergebnisse werden mit den Ergebnissen ohne Granularitätswechsel verglichen.

Die Evaluation der Ergebnisse findet sich in Kapitel 5. Detaillierte Ergebnisse, die die Übersichtlichkeit von Kapitel 5 einschränken würden, finden sich im Anhang (Kapitel A).

3 Verwandte Arbeiten

Der Teilbereich des Argument Minings stellte in den letzten Jahren vermehrt die Grundlage vieler Arbeiten dar. Dabei wurden Daten aus unterschiedlichen Quellen annotiert, wie Gesetzestexte (Mochales und Moens, 2011, Feng und Hirst, 2011), Produkt bzw. Hotelbewertungen (Schneider und Wyner, 2012, Wachsmuth et al., 2014), Online-Diskussionen zu erneuerbaren Energien (Goudas et al., 2014) oder zu kontroversen Themen im Bildungssystem (Habernal und Gurevych, 2017). Ebenfalls wurden einige Teilbereiche des Argument Minings genauer untersucht, so z.B. in Habernal und Gurevych (2016) die Aufgabe zu erkennen, welches von zwei Argumenten überzeugender ist. Eine Übersicht über einige der einflussreichsten Arbeiten der letzten Jahre im Bereich des Argument Minings findet sich in Habernal und Gurevych (2017). Die jeweils untersuchten Klassifikationsaufgaben im Bereich des Argument Minings wurden dabei sowohl durch klassische Machine-Learning Algorithmen wie SVMs oder naive Bayes Klassifkatoren, als auch durch neuronale Netze wie BLSTMs (Habernal und Gurevych, 2016) gelöst.

Eine im Speziellen interessante Arbeit durch ihre Untersuchung von Online-Diskussionen, Verwendung eines ähnlichen Annotationsschematas sowie die Untersuchung auf feiner Granularitätsebene stellt dabei Habernal und Gurevych (2017) dar. Dort werden Online-Diskussionen zu Bildungsthemen wie Hausunterricht, private gegen öffentliche Schulen oder getrennter Jungen- und Mädchenunterricht annotiert und sowohl aus einer sozialwissenschaftlichen Perspektive hinsichtlich Argumentationen als auch aus technischer Perspektive bezüglich der Leistung von Klassifikatoren und Nutzbarkeit von Features analysiert. Dadurch weist die Arbeit Ähnlichkeiten zu den Untersuchungen in Liebeck et al. (2016), Liebeck et al. (2017) und Liebeck (2018) auf. Ebenfalls ähnlich zu dieser Arbeit ist die Verwendung eines auf Online-Argumentation angepassten Argumentationsschemas, welches im Falle von Habernal und Gurevych (2017) auf dem in Kapitel 1 beschriebenen Modell von Toulmin basiert. Habernal und Gurevych (2017) modifizierten dabei das Modell unter Anderem dahingehend, dass „Qualifiers“ und „Warrants“ entfernt werden, Angriffe auf „Rebuttals“ mit dem Namen „Refutation“ eingeführt werden und die in Toulmins Modell als „Grounds“ bezeichneten Argumente im Sinne vorangehender Arbeiten in „Premises“ umbenannt werden. Habernal und Gurevych (2017) verwenden dabei Daten aus Kommentaren zu Zeitungsartikeln, Blog-Beiträgen sowie Foren, welche sich hinsichtlich der verwendeten Formulierungen, Rechtschreibung, inhaltlichen Präzision sowie Länge deutlich unterscheiden (Habernal und Gurevych, 2017, S. 139 u. S. 145) und auf diese Weise ähnlich große Differenzen bezüglich der Form der untersuchten Texte aufweisen, wie die beiden in dieser Arbeit untersuchten Korpora THF und Braunkohle. Insgesamt wurden von Habernal und Gurevych (2017) 990 Dokumente (Beiträge, Kommentare etc.) annotiert, wovon 524 als argumentativ eingeordnet wurden.

Eine der stärksten Gemeinsamkeiten und Motivation zur Untersuchung von verschiedenen Granularitäten stellt dabei Habernal und Gurevych (2017) und deren Ansatz des Sequence Taggings dar. Hierbei verwenden Habernal und Gurevych (2017) Tokens als kleinste Annotationseinheit eines Sequence Labeling Problems, welches aufgrund der semantischen Struktur der verwendeten Variablen auch auf diese Arbeit zutrifft, da trotz der Möglichkeit der Annotation auf Zeichenebene inhaltlich meist ganze Tokens codiert wurden. Habernal und Gurevych (2017) nutzen für die Codierung der Sequenzkompo-

nenten das BIO-Schema (Ramshaw und Marcus, 1999). Dieses Schema unterscheidet jede Variable hinsichtlich ihrer Position in einer zusammenhängend codierten Sequenz. Hierbei repräsentiert B (Beginning) die erste codierte Einheit in der Sequenz, I (Inner) alle weiteren Einheiten in der Sequenz und O (Outside) alle Einheiten, die außerhalb der Sequenz liegen. Auf diese Weise können auch Sequenzen die mit gleicher codierter Klasse nebeneinander liegen unterschieden werden.

Aufgrund der in Habernal und Gurevych (2017) vorgenommenen Annotation auf Satzebene mit nur wenigen Ausnahmen, die die Korrektur auf Tokenebene erforderte, verwenden Habernal und Gurevych (2017) allerdings für die Klassfikationsaufgaben ein vereinfachtes Modell, welches die potenziell auf Tokenebene vorliegenden Annotationen mit Hilfe eines Mehrheitsentscheids auf Satzebene reduziert und dann die eigentliche Klassifikation auf Satzebene durchführt. Anschließend werden die so klassifizierten Label auf Satzebene wieder auf Tokenebene transformiert, siehe Abbildung 3. Aufgrund dieser Einschränkung der Klassifkationsgranularität in Habernal und Gurevych (2017) bleiben der Vergleich und die Auswertung der Anwendbarkeit von Klassifikationsaufgaben mit feinerer Granularität vor dem Hintergrund von Argument Mining in Online-Diskussionen ein interessantes Thema. Goudas et al. (2014) beschreiben einen mehrschichtigen Klassifikationsprozess in dem nach einem ersten Schritt, in dem argumentative Sätze erkannt werden, ein zweiter Schritt diejenigen Teile der Sätze feingranular klassifiziert, welche die jeweiligen Argumente enthalten. Hier konnten unter anderem gute Ergebnisse für SVMs im ersten Schritt und für Conditional Random Fields im zweiten Schritt erzielt werden. Diese Ergebnisse motivieren die Untersuchung dieser Klassifikatoren in dieser Arbeit. Eine weitere Arbeit mit ähnlichem Hintergrund stellt Guggilla et al. (2016) dar. Hier wurden gute Ergebnisse für die Klassifikation von Argumenten in Online-Diskussionen durch Verwendung von CNN und LSTMs erzielt. Diese konnten dort ähnlich gute Ergebnisse liefern wie SVMs mit reichhaltigen Features.

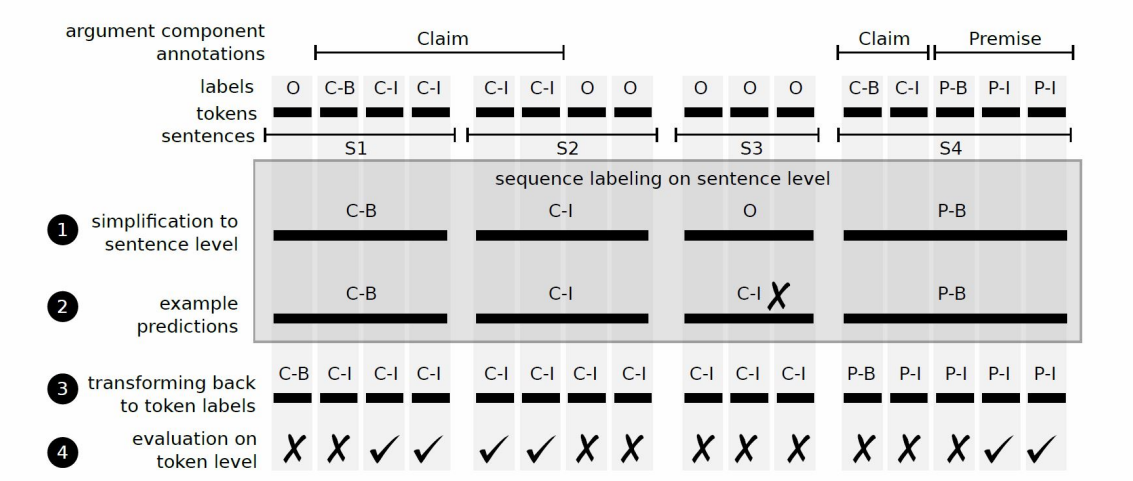


Abbildung 3: Veranschaulichung der in Habernal und Gurevych (2017) vorgenommen Abstraktion der Klassifkation von Tokenebene auf Satzebene. C-B, C-I, P-B, P-I und O stehen dabei für die Labelnamen der Klassen Claim und Premise nach dem verwendeten BIO-Schema. Entnommen aus Habernal und Gurevych (2017, S.158).

4 Methoden

Im Folgenden werden die verwendeten Konzepte und Implementierungen der in dieser Arbeit vorgenommen Klassifkationen behandelt.

4.1 Framework

Da diese Arbeit auf Liebeck (2018) aufbaut und damit vergleichbare Resultate auf den in Kapitel 2 beschriebenen neuen Korpora liefern soll, wurde das in Liebeck (2018) entwickelte Klassifikationsframework verwendet und um neue Funktionalitäten, wie der Verarbeitung von verschiedenen Granularitäten, erweitert. Zudem wurden einige alternative Klassifikatoren außerhalb des Frameworks implementiert.

Das Framework von Liebeck (2018) wurde in Python entwickelt und verwendet die auf Machine Learning und Natural Language Processing ausgerichteten Bibliotheken spaCy[8] sowie scikit-learn (Pedregosa et al., 2011). Die Implementierungen der einzelnen Machine Learning Algorithmen sowie der in scikit-learn verwendeten Pipeline und anderen Workflow-Komponenten werden dabei genutzt, um ein modulares System zur Evaluierung verschiedener Klassfikatoren und Features zu erhalten. Im Fall der implementierten klassischen Machine-Learning Algorithmen wurde zudem aufgrund der großen Menge an Parametern ein Gridsearch-Verfahren zur Ermittlung der besten Parameter implementiert. Aufgrund der Menge an Parameterkombinationen wurde dieser Teil ebenfalls für die parallele Ausführung auf dem HPC-Cluster Hilbert der Universität Düsseldorf[9] ausgelegt.

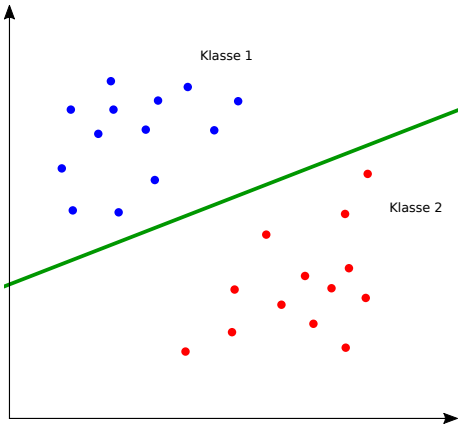
Der grundsätzliche Aufbau des Frameworks lässt sich dabei wie folgt beschreiben: Das Projekt ist in vier Teile aufgeteilt: Daten, Experimente, Frameworks und Tests. Der Datenteil umfasst alle für die Ausführung der Experimente notwendigen Daten. Dazu gehören die in passenden Formaten vorliegenden Korpora sowie vortrainierte Wortrepräsentationen (z.B. word2vec (Mikolov et al., 2013b), LDA (Blei et al., 2003) etc.). Der Framework-Teil beinhaltet Klassifikatoren, Features sowie Evaluationsmetriken. Die einzelnen Komponenten dieses Teils lassen sich dabei in die Pipeline-Architektur von sklearn integrieren und werden im Testteil mit Hilfe von Unittests validiert. Der Experimente-Teil beinhaltet schließlich die nötigen Skripte zur Ausführung des Gridsearch (im Falle der klassischen Machine-Learning Algorithmen) sowie der Evaluation in Form einer mehrfachen Überkreuzvalidierung. Dieser Teil beinhaltet auch die Skripte zur Erstellung der für den HPC Cluster nötigen PBS-Jobfiles.

Aufgrund

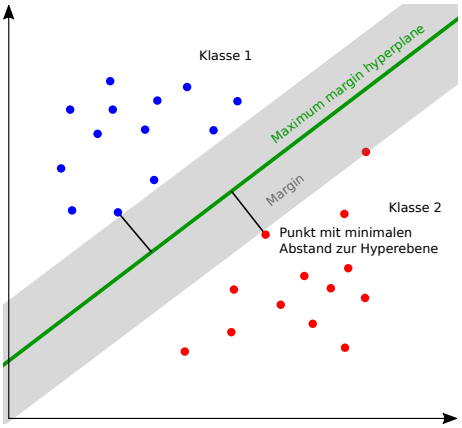
4.2 Klassisches Machine Learning

Der klassische Machine Learning Teil umfasst Implementierungen von Support Vector Machines (SVM), Random Forests (RF) sowie K-Nearest Neighbor Klassfikatoren (KNN). Aufgrund der schlechten Ergebnisse des KNN-Klassifkators in Liebeck (2018) und der

[8] https://spacy.io/
[9] https://www.zim.hhu.de/high-performance-computing.html



(a) Beispiel für die lineare Separation der Daten durch eine Hyperebene (hier: grüne Linie) in zwei Klassen



(b) Beispiel einer 2D-SVM mit *maximum margin hyperplane*

Abbildung 4: Beispiele für SVM-Konstruktion

Anzahl an zu testenden Klassifkator-Feature Kombinationen wurde dieser Klassifkator hier nicht weiter getestet. Zusätzlich wurden außerhalb des Frameworks Experimente zu Conditional Random Fields (Lafferty et al., 2001), insbesondere für die feingranulare Klassifkation auf Tokenebene, durchgeführt. Dazu wurden die Bibliotheken Python-crfsuite[10] bzw. deren Implementierungen für scikit-learn in sklearn-crfsuite[11] verwendet.

4.2.1 Support Vector Machine

In dieser Arbeit werden unter anderem *Support Vector Machines* nach Cortes und Vapnik (1995) zur Klassifikation der Daten verwendet. Klassischerweise werden SVMs dabei für binäre Klassifikationsaufgaben eingesetzt, können aber auch auf Mehrklassenprobleme ausgeweitet werden, indem entweder für jedes Klassenpaar „1-gegen-1“ verglichen wird oder aber jede Klasse gegen die restlichen Klassen („1-gegen-Rest“) verglichen wird (Milgram et al., 2006).

Support Vector Machines basieren auf der Idee, die vorliegenden Daten als Vektoren im Raum linear zu separieren, um damit eine Klassifikation zu ermöglichen. Das Prinzip dabei ist es eine solche Hyperebene zu finden, welche den Datenraum so teilt, dass alle Daten einer Klasse auf einer Seite der Ebene liegen und alle Daten der anderen Klasse auf der anderen wobei der Abstand der Hyperebene zu den Datenpunkten auf beiden Seiten maximiert wird. Weitere Punkte können dann anhand ihrer Position zu dieser Hyperebene, d.h. ober- oder unterhalb dieser, einer der beiden Klassen zugeordnet werden, siehe Abbildung 4.

Um eine möglichst robuste Klassifikation zu erreichen, ist es dabei wichtig die linear separierende Hyperebene optimal zu wählen. D.h. sie sollte zu den gegebenen Daten einen möglichst hohen minimalen Abstand aufweisen. Eine solche Ebene wird dann *maximum*

[10] https://python-crfsuite.readthedocs.io/en/latest/
[11] https://sklearn-crfsuite.readthedocs.io/en/latest/

margin hyperplane genannt und kann mit Hilfe von Optimierungsverfahren für das, hier anschaulich vereinfachte, Problem

$$\text{Maximiere: } \xi = \min_{\text{Punkt} \in \text{Trainingsmenge}} |\text{Abstand}(\text{Punkt}, \text{Hyperebene})|$$

unter der Berücksichtigung, dass die Daten auf der korrekten Seite der Ebene liegen, gefunden werden (Siehe Abbildung 4). Da hierbei die *maximum margin hyperplane* nur durch Vektoren (hier Stützvektoren bzw. *support vectors*), welche auf den um die *margin* verschobenen parallelen Hyperebenen liegen, bestimmt wird, ergibt sich der Name Support Vector Machine. Diese Vektoren stellen anschaulich diejenigen Punkte mit geringstem Abstand zur *maximum margin hyperplane* dar (siehe auch Abbildung 4). Formal lässt sich dieses Problem wie folgt beschreiben (angelehnt an Cortes und Vapnik (1995)):

Seien $x_i, i \in \{1, \ldots, r\}$ die Trainingsdatenpunkte aus dem Datenraum FS, die den Labeln $y_i \in \{-1, 1\}$ zugeordnet sind. Für einen Vektor $w \in \mathbb{R}^n$ und ein Skalar $b \in \mathbb{R}$ wird eine Hyperebene durch

$$H(w, b) = \{x \in FS | w \cdot x + b = 0\} \tag{1}$$

beschrieben und der Abstand eines Vektors x zu dieser Ebene ist durch

$$dist(x, H(w, b)) = \left| \frac{w}{\sqrt{w \cdot w}} \cdot x + b \right| \tag{2}$$

definiert. Die Trainingsdatenpunkte werden hierbei als linear separierbar bezeichnet wenn ein Vektor $w \in \mathbb{R}^n$ und Skalar $b \in \mathbb{R}$ existieren sodass folgende Ungleichungen für alle $i \in \{1, \ldots, r\}$ gelten:

$$\begin{aligned} w \cdot x_i + b \geq 1 \quad &\text{falls} \quad y_i = 1 \\ w \cdot x_i + b \leq -1 \quad &\text{falls} \quad y_i = -1 \end{aligned} \tag{3}$$

Dies kann zusammengefasst werden zu:

$$y_i(w \cdot x_i + b) \geq 1, \quad i = 1, \ldots, l. \tag{4}$$

Die optimale Hyperebene

$$w_0 \cdot x + b_0 = 0 \tag{5}$$

ist diejenige Hyperebene mit Normalenvektor $w_0 \in \mathbb{R}^n$ und Skalar $b_0 \in \mathbb{R}$, welche die Daten mit maximalem Rand (*margin*) separiert. Sie bestimmt die Richtung $\frac{w}{|w|}$ nach der die Projektionen der Trainingsvektoren von zwei verschiedenen Klassen den größten Abstand aufweisen. Diese Distanz $\rho(w, b)$ ist gegeben durch:

$$\rho(w, b) = \min_{\{x:y=1\}} \frac{x \cdot w}{|w|} - \max_{\{x:y=-1\}} \frac{x \cdot w}{|w|} \tag{6}$$

Die optimale Hyperebene (w_o, b_o) optimiert dabei die Distanz aus Gleichung 6. Aus Gleichungen 4 und 6 folgt dann:

$$\rho(w_0, b_0) = \frac{2}{|w_0|} = \frac{2}{\sqrt{w_0 \cdot w_0}} \tag{7}$$

Das primäre Maximierungsproblem maximiert somit

$$\frac{2}{\sqrt{w_0 \cdot w_0}} \tag{8}$$

Dieses Problem lässt sich zu einem Minimierungsproblem umformen:

$$\text{Minimiere: } J(w, b) = |w| \tag{9}$$

$$\text{unter Nebenbedingung: für } i = 1, \ldots, n \text{ sei } (y_i \cdot (w \cdot x_i + b) \geq 1) \tag{10}$$

Da die zu klassifizierenden Daten trotz binärer Klassifikationsaufgabe nicht immer linear separarierbar sind, werden die Daten häufig vor der Anwendung der SVM mittels einer Kernelfunktion auf einen erweiterten Raum abgebildet, wodurch die Daten in einer besser separierbaren Form vorliegen. Einer der gängigsten Kernel ist dabei der *radial basis function*-Kernel. Zudem kann auf die strikte Einhaltung der Klassifikation der Trainingsdaten verzichtet werden, indem ein tolerierbarer Fehler für die Positionierung der Hyperebene eingeführt wird. Anschaulich heißt das, dass einige der Trainingsdaten (im Idealfall Ausreißer) auf der falschen Seite der Ebene liegen um eine größere *margin* zu erreichen. Diese Methode wird auch als *soft-margin*-Optimierung bezeichnet (Cortes und Vapnik, 1995).

Im Falle dieser Arbeit werden zwei verschiedene Arten von SVMs verwendet. Eine lineare SVM, die der klassischen Definition entspricht, sowie eine SVM mit RBF-Kernel. Beide Varianten wurden durch den Support Vector Klassifikator aus der Sklearn-Bibliothek[12] umgesetzt.

Aufgrund der Inkompatibilität der verwendeten Bibliotheken für Conditional Random Fields[13][14] mit dem bestehenden Framework, welche ein Integration nur mit großen Kompromissen ermöglicht hätte, wurde für diesen Klassifikator die Funktionalität des Frameworks in einem separaten Projekt nachgeliefert.

4.2.2 Entscheidungsbäume und Random Forest

Entscheidungsbäume (Quinlan, 1986) sind Bäume, die im Kontext von Klassifikationsaufgaben durch aufeinanderfolgende Entscheidungen zwischen Attributsausprägungen Datenpunkte klassifizieren können. Dazu repräsentiert jeder innere Knoten eines Entscheidungsbaumes ein Attribut. Ein solches Attribut könnte beispielsweise für die Klassifikationsaufgabe „Spazierengehen? Ja:Nein“ das kategorische Attribut „Wetteraussicht“ mit Ausprägungen „sonnig“ „bedeckt“ und „regnerisch“ sein. Ausgehend von der Wurzel des Baumes wird nun für einen zu klassifizierenden Datenpunkt bei jedem Knoten (und somit Attribut) geprüft, welche Attributausprägung für den Datenpunkt zutrifft. Der Entscheidungsbaum besitzt nun am aktuellen Knoten für jede Ausprägung eine ausgehende Kante, entlang welcher zum nächsten zu testenden Attribut gefolgt wird. Die Blätter des Baumes enthalten dann die Label der Klassifikationsaufgabe. Die Klassifikation eines Datenpunktes erfolgt somit, sobald während der Traversierung des Baumes auf

[12] http://scikit-learn.org/stable/
[13] https://python-crfsuite.readthedocs.io/en/latest/
[14] https://sklearn-crfsuite.readthedocs.io/en/latest/

Tag	Aussicht	Temperatur	Feuchtigkeit	Wind	Spazierengehen
1	sonnig	heiß	hoch	schwach	nein
2	sonnig	heiß	hoch	stark	nein
3	bedeckt	heiß	hoch	schwach	ja
4	regnerisch	mild	hoch	schwach	ja
5	regnerisch	kühl	normal	schwach	ja
6	regnerisch	kühl	normal	stark	nein
7	...	...	...	...	...

Tabelle 13: Kategorische Beispieldaten für die Verwendung in einem Entscheidungsbaum. Die verwendeten kategorischen Attribute Aussicht, Temperatur, Feuchtigkeit und Wind sollen für die Klassifikation der Tage in Spazierengehen-ja und Spazierengehen-nein verwendet werden. Angelehnt an Quinlan (1986) und das KDD-Skript von Prof. Dr. Stefan Conrad (HHU) (Conrad, 2016).

ein Blatt gestoßen wird. Eine Veranschaulichung eines Entscheidungsbaumes findet sich in Tabelle 13 und Abbildung 5.

Die Konstruktion eines solchen Baumes erfolgt über eine Reihe von Splits. Am Anfang werden alle Trainingsdatensätze der Wurzel zugeordnet. Nun wird anhand einer Splitstrategie das nächste Attribut ausgewählt, welches für den Split verwendet werden soll. Ist ein solches Attribut gefunden, werden die Trainingsdaten durch das Splitattribut partitioniert. Dieses Verfahren wird dann rekursiv auf die neu enstandenen Partitionen angewendet bis es entweder keine ungenutzten Splitattribute mehr gibt oder die in diesem Schritt untersuchte Partition bereits nur Datensätze einer Klasse enthält.

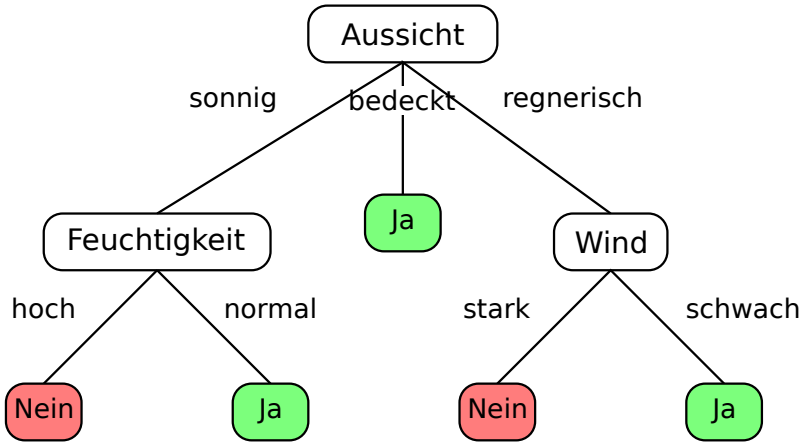


Abbildung 5: Beispiel für einen Entscheidungsbaum für die Klassifikation von Wettern bezüglich Spazieren gehen-ja und -nein. Innere Knoten repräsentieren Attribute, Blätter geben die Klassen an. Angelehnt an Quinlan (1986) und das KDD-Skript von Stefan Conrad (HHU) (Conrad, 2016).

Für die Bestimmung des für einen Split zu verwendenen Attributes muss dabei ein Gütemaß für Splits eingeführt werden. Gewünscht ist dabei ein Maß, welches die Unreinheit,

d.h. Vermischung, der Klassen in den durch den Split entstandenen Partitionierungen misst. Dies kann durch Gütemaße wie dem Informationsgewinn oder Gini-Index erreicht werden. Formal kann das Problem dabei wie folgt beschrieben werden: Seien X die Menge aller Trainingsdaten, $c_1, c_2, \ldots, c_k$ die verschiedenen Klassen und $p_i, i \in \{1, \ldots, k\}$, die relativen Häufigkeiten der Klassen c_i in X. Dann wird eine vollständige, disjunkte Partitionierung von X in $X_1, X_2, \ldots, X_m$ gesucht, für welche die Unreinheit dieser Partitionierung minimiert wird.

Zur Bestimmung der Unreinheit können Entropie/Informationsgewinn bzw. Gini-Index verwendet werden. Die Entropie H einer Menge X ist dabei wie folgt definiert:

$$H(X) = -\sum_{i=1}^{k} p_i \cdot \log_2 p_i \tag{11}$$

Hierbei wird $0 \cdot \log 0 = 0$ gesetzt. Es ist ersichtlich, dass für eine reine Datenmenge, d.h. für eine Menge mit $p_i = 0, \forall i \in \{1, \ldots, k\}$ und einem $p_{i_0} = 1$, gilt: $H(X) = 0$ und für eine komplett unreine Datenmenge, z.B. für zwei Klassen und beiden $p_i = \frac{1}{2}$, dass $H(X) = 1$. Somit kann die Entropie als Maß für die Unreinheit einer Datenmenge verwendet werden. Für die Bestimmung der Güte einer Partitionierung wird darauf aufbauend der Informationsgewinn verwendet. Der Informationsgewinn I einer Datenmenge X bezüglich der Partitionierung, die sich durch Verwendung des Attributes A für die Bildung des Splits ergibt, ist formal definiert als:

$$I(X, A) = H(X) - \sum_{i=1}^{m} \frac{|X_i|}{|X|} \cdot H(X_i) \tag{12}$$

Ähnlich zur Entropie kann auch der Gini-Index G für eine Datenmenge X definiert werden:

$$G(X) = 1 - \sum_{i=1}^{k} p_i^2 \tag{13}$$

Auch hier gilt, dass ein kleiner Gini-Index eine kleine Unreinheit und ein großer Gini-Index eine große Unreinheit der Daten anzeigt. Für eine Partitionierung von X durch das Attribut A wird der Gini-Index G dann wie folgt definiert:

$$G(X, A) = \sum_{i=1}^{m} \frac{|X_i|}{|X|} \cdot G(X_i) \tag{14}$$

Hierbei gilt allerdings anders als im Fall des Informationsgewinns, dass ein kleinerer Wert von G eine geringere Unreinheit (z.B. für ein $p_i = 1 \Rightarrow G = 0$) und ein größerer Wert (maximal $\frac{1}{2}$ für zwei Klassen mit $p_1 = p_2 = 0,5$) eine größere Unreinheit repräsentiert.

Für die Bestimmung eines Splitattributes kann nun der Gini-Index oder Informationsgewinn genutzt werden, um dasjenige Attribut auszuwählen, welches die Unreinheit der sich ergebenen Partitionierung minimiert. Dabei ist zu beachten, dass hierbei immer nur eine lokale Optimierung stattfindet. Um den bestmöglichen Baum zu bestimmen, müssten alle Kombinationen von Splits durchsucht werden, da nur so sichergestellt werden kann, dass ein globales Optimum gefunden wird. Da ein solche Vorgehen sehr rechenintensiv ausfällt werden meist heuristische Methoden zur Ermittlung eines lokalen Optimums verwendet. Ein weiterer Nachteil von klassischen Entscheidungsbäumen stellt die

starke Abhängigkeit des Baumes von den verwendeten Trainingsdaten dar. Schon kleine Änderungen in den Trainingsdaten können zu einer anderen Baumstruktur führen, sodass das Verfahren leicht anfällig für Overfitting wird.

Eine Lösung für dieses Problem stellt das Random Forest Verfahren nach Ho (1995) dar. Mit Random Forest wird dabei ein Wald aus Entscheidungsbäumen bezeichnet, welcher auf Untermengen des Trainingsdatensatzes trainiert wurde. Die Klassifikation erfolgt dann durch alle einzelnen Entscheidungsbäume durch Mehrheitsentscheid. Eine Besonderheit des Random Forest gegenüber anderen Ensemble-Verfahren auf Basis von Entscheidungsbäumen ist dabei allerdings die zusätzliche, zufällige Einschränkung der benutzen Attribute bei jedem Entscheidungsbaum. Da Entscheidungsbäume leicht zu Overfitting neigen und dabei stark entscheidende Attribute in allen Bäumen ähnliche Effekte haben würden, werden daher bei l Attributen etwa $\sqrt{l}$ zufällige Attribute je Entscheidungsbaum verwendet.

Für die Implementierung wird in dieser Arbeit der Random Forest Ensemble Klassifikator von Sklearn (*sklearn Random Forest* o.D.) verwendet.

4.2.3 Conditional Random Fields

Conditional Random Fields stellen ein probabilistisches Modell dar, welches ähnlich zu Hidden Markov Modellen und logistischer Regression verstanden werden kann, und sich insbesondere für die Klassifikation von Sequenzen eignet. Für das Verständnis der Gemeinsamkeiten folgt eine kurze Beschreibung von HMM und Logistischer Regression.

Ein Hidden Markov Modell λ kann formal als Quintupel aufgefasst werden:

$$\begin{aligned}
&\lambda = (S, V, A, B, \pi) \\
&S = \{s_1, s_2, \ldots, s_N\}, N \in \mathbb{N} \text{ (Menge aller Zustände)} \\
&V = \{v_1, v_2, \ldots, v_M\}, M \in \mathbb{N} \text{ (Menge aller Beobachtungen)} \\
&A \in \mathbb{R}^{N \times N} \text{ (Übergangsmatrix)} \\
&B \in \mathbb{R}^{N \times M} \text{ (Beobachtungsmatrix)} \\
&\pi \in \mathbb{R}^{N} \text{ (Anfangsverteilung)}
\end{aligned} \tag{15}$$

Werden hierbei die Zustände als Klassen und Beobachtungen als Attribute aufgefasst, kann mithilfe von Algorithmen wie dem Viterbi-Algorithmus (Viterbi, 1967), welche die wahrscheinlichste Zustandsreihenfolge bei gegebener Beobachtungssequenz bestimmen, ein HMM für Klassifikationsaufgaben verwendet werden.

Logistische Regression stellt ein statistisches Modell dar, welches in seiner einfachsten Form zur binären Klassifikation eingesetzt werden kann. Formal kann dies wie folgt beschrieben werden (angelehnt an Abu-El-Haija, 2018): Sei $X = \{(x_1, l_1), (x_2, l_2), \ldots, (x_N, l_N)\}$ ein Datensatz mit Datenpunkten $x_i \in \mathbb{R}^M, i \in \{1, \ldots, N\}$, und Klassen $l_i \in \{0, 1\}, i \in \{1, \ldots, N\}$. Dann erstellt die logistische Regression ein probabilistisches Modell der beiden Klassen $p(l = 1|x, w)$ und $p(t = 0|x, w)$, wobei $w \in \mathbb{R}^M$ den durch Maximum-Likelihood-Schätzung zu optimierenden Gewichtungsparameter darstellt. Das Modell selbst berechnet dabei die Wahrscheinlichkeit der Klassenzugehörigkeit, z.B. für die Klassen $l = 1$ bzw. $l = 0$, wie

folgt:

$$\begin{aligned} p(l = 1, x, w) &= \sigma(w^\intercal x) \\ p(l = 0, x, w) &= 1 - p(l = 1, x, w) = 1 - \sigma(w^\intercal x) \end{aligned} \tag{16}$$

, wobei $\sigma(t) = \frac{1}{1+\exp(-t)}$ die Sigmoidfunktion darstellt. Anschaulich muss nun ein w gefunden werden, welches unter den für die Zuordnung relevanten Komponenten eines Datenpunktes denjenigen Komponenten, die zu Klasse 1 gehören, ein hohes positives Gewicht gibt und denjenigen aus Klasse 0 ein negatives Gewicht gibt. Zur Bestimmung des optimalen w wird dabei die Maximum-Likelihood-Schätzung verwendet. Dazu wird dasjenige w gesucht, welches die Likelihood-Funktion $L(w)$ maximiert:

$$L(w) = \prod_{i=1}^{N} p(l_i | x_i, w) \tag{17}$$

Zur einfacheren Berechnung wird hierbei die Log-Likelihood Funktion verwendet:

$$\arg\max_w L(w) = \arg\max_w \log L(w) \tag{18}$$

, wobei erneut $\log 0 = 0$ gesetzt wird. Für die Lösung dieses Problems werden, da es sich um ein Maximierungsproblem handelt, meist Gradientenaufstiegsverfahren verwendet.

Es folgt eine grobe Beschreibung der Funktionsweise von CRFs, angelehnt an Chen (2018a).

Conditional Random Fields (Lafferty et al., 2001) werden häufig zur Klassifikation von Sequenzen verwendet. Dies können insbesondere Anwendungen im NLP Bereich, wie z.B. das Part-of-Speech Tagging, oder wie in dieser Arbeit das Argument Mining, sein. Aufgrund ihrer Struktur verwenden Conditional Random Fields Kontextinformationen durch Berücksichtigung der zuvor erhaltenen Label in einer Sequenz. Sie eignen sich daher gut für NLP-Anwendungen, in denen auch semantisch eine hohe Abhängigkeit der Komponenten einer Sequenz (z.B. der Kasus von Wörtern eines Satzes) besteht.

Für die Konstruktion von Conditional Random Fields ist dabei die Definition von Feature-Funktionen wichtig. Im Falle der in dieser Arbeit verwendeten Linear-Chain CRF handelt es sich dabei um Funktionen, die für jeden Datenpunkt in einem Dokument (z.B. ein Token in einem Satz im Falle der Satzebene) als Eingabe das Dokument, die Position eines Datenpunktes im Dokument, die Klasse des aktuellen Datenpunktes sowie des vorherigen Datenpunktes erwarten. Für das Beispiel auf Satzebene wären dies ein Satz s, die Position i eines Wortes in s sowie die Klassen l_i des aktuellen Wortes und l_{i-1} des vorherigen Wortes. Eine Featurefunktion f_j berechnet dann:

$$f_j(s, i, l_i, l_{i-1}) \in \mathcal{R} \tag{19}$$

, wobei das Ergebnis meist auf den Bereich zwischen 0 und 1 normiert wird. Jeder Featurefunktion f_j wird zusätzlich ein Gewicht λ_j zugeordnet. Nun kann für eine mögliche Klassifikation durch die Labelsequenz l ein Score berechnet werden:

$$score(l|s) = \sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s, i, l_i, l_{i-1}) \tag{20}$$

Diese Scores können in Wahrscheinlichkeiten $p(l|s)$ zwischen 0 und 1 umgewandelt werden:

$$p(l|s) = \frac{\exp(score(l|s))}{\sum_{l'} \exp(score(l'|s))} = \frac{\exp(\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s, i, l_i, l_{i-1}))}{\sum_{l'} \exp(\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(s, i, l'_i, l'_{i-1}))} \tag{21}$$

Hierbei werden die Scores einer Labelsequenz l im Verhältnis zu den Scores aller anderen möglichen Labelsequenzen l' betrachtet.

Ein Beispiel für eine Feature-Funktion (angelehnt an Chen, 2018a) wäre $f_1(s, i, l_i, l_{i-1}) = 1$ falls $i = 1$ und l_i=VERB und der der Satz mit einem Fragezeichen endet. Sonst ist $f_1(s, i, l_i, l_{i-1}) = 0$. Ist das zugehörige λ_1 groß, werden solche Klassifikationen bevorzugt, in denen das erste Wort eines Satzes als Verb klassifiziert wird, wenn der Satz mit einem Fragezeichen endet.

Um die Gewichte λ_j zu optimieren wird nun ähnlich zur logistischen Regression ein Gradientenaufstiegsverfahren bezüglich der Log-Wahrscheinlichkeiten verwendet: Für jede Featurefunktion f_i wird der Gradient bezüglich λ_j der Log-Wahrscheinlichkeit gebildet:

$$\frac{\partial}{\partial \lambda_j} \log p(l|s) = \sum_{j=1}^{m} f_i(s, j, l_j, l_{j-1}) - \sum_{l'} p(l'|s) \sum_{j=1}^{m} f_i(s, j, l'_j, l'_{j-1}) \tag{22}$$

Hierbei stellt die erste Summe den Beitrag von f_i mit korrekter Klasse und die zweite Summe den erwarteten Beitrag von f_i im aktuellen Modell dar. Das Gewicht λ_i wird dann in jedem Lernschritt mit Lernrate α zu λ'_i aktualisiert:

$$\lambda'_i = \lambda_i + \alpha(\sum_{j=1}^{m} f_i(s, j, l_j, l_{j-1}) - \sum_{l'} p(l'|s) \sum_{j=1}^{m} f_i(s, j, l'_j, l'_{j-1})) \tag{23}$$

Um nun mit Hilfe eines trainierten CRF-Modells die Wörter eines Satzes zu klassifizieren, müsste für jede mögliche Labelsequenz l die Wahrscheinlichkeit $p(l|s)$ gebildet werden. Da dies bei einem Satz der Länge m mit k möglichen Klassen zu k^m Labelsequenzen führen würde, wird in der Praxis meist ein Verfahren zur Findung von optimalen Lösungen von Unterproblemen, ähnlich dem in HMMs verwendeten Viterbi-Algorithmus, eingesetzt. Dies sowie die Möglichkeit zu jeder HMM eine equivalente CRF zu konstruieren (Chen, 2018a), zeigt auch die Ähnlichkeiten von CRFs und HMMs auf. In dieser Arbeit werden nur linear-chain CRFs verwendet, welche ihren Namen durch ihre mögliche Repräsentation als linearer Pfad in einer Graphstruktur (ähnlich zu HMMs) erhalten. Dies spiegelt sich auch in den Featurefunktionen wieder, welche immer das aktuelle und vorherige Label als Input auffassen. Im Gegensatz dazu können CRFs allgemein als ein Graph mit beliebigen Kanten aufgefasst werden, sodass auch die Featurefunktionen nicht auf zwei aneinander grenzende Label beschränkt sind. Weitere Informationen dazu können in Sutton, McCallum et al. (2012) gefunden werden.

Für die Implementierung von Conditional Random Fields wurden die Bibliotheken python-crfsuite[15] und sklearn-crfsuite[16] verwendet.

[15] https://python-crfsuite.readthedocs.io/en/latest/
[16] https://sklearn-crfsuite.readthedocs.io/en/latest/

4.3 Deep Learning: Neuronale Netze

Neuronale Netze stellen eine andere Familie von Verfahren dar, welche zur Klassifikation von Daten verwendet werden können. Der Aufbau dieser Netze orientiert sich grob an dem gegenwärtigen Verständnis der Vorgänge im menschlichen Gehirn und nutzen eine Graphstruktur, in der die Funktionsweise von *Neuronen* nachempfunden wird. Hierbei werden die als *Neuronen* bezeichneten Knoten des Graphen in Schichten organisiert und durch Kanten verbunden. Eine erste Schicht wird dabei als *Input-layer* bezeichnet, die letzte Schicht als *Output-layer*, während alle Schichten dazwischen unter den Oberbegriff *Hidden-layer* fallen.

Abbildung 6 zeigt ein vereinfachtes Modell eines einfachen neuronalen Netzes (*Feed-forward*-Netz mit Hidden layer). Hierbei wurden einige Neuronen und zusätzliche Werte exemplarisch ohne Anspruch auf Vollständigkeit eingetragen. Die Werte $w1, w2$ und $w3$ geben dabei Gewichte an, $b6$ einen sogenannten *Bias* und f die Aktivierungsfunktion von Neuron $N6$. Das Prinzip eines neuronalen Netzes ist es nun in einem *Forward propagation*-Schritt alle Inputs, welche den Neuronen des *Input-layers* zugewiesen werden, durch das Netz bis hin zu den Output-Neuronen zu propagieren. Dabei ergibt sich der Wert (Aktivität) eines Neurons durch die Eingaben aller mit ihm verbundenen Neuronen der vorherigen Schicht sowie häufig einem zusätzlichen fixen *Bias*-Wert. Die gewichtete Summe der eingehenden Kanten wird dann in ein Signal zwischen 0 und 1 mittels einer Aktivierungsfunktion, z.B. der Sigmoidfunktion

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{24}$$

, überführt und bestimmt somit den Output dieses Neurons.

Im Beispiel von Abbildung 6 würde sich somit die Aktivität von $N6$ durch die Summe der durch $w1, w2$ und $w3$ gewichteten Inputs von $N1, N2$ und $N3$ mit dem *Bias* $b6$ sowie der Anwendung der Aktivierungsfunktion f ergeben. Diese Aktivität von $N6$ würde dann, verändert durch eine Outputfunktion (meist die Identitätsfunktion), wiederum für die Berechnung von $N9, N10$ und $N11$ verwendet werden, die dann zur Berechnung der Aktivität von $N14$ und $N15$ beitragen.

Eine mögliche Art des Trainings eines solchen Systems besteht darin, die Daten der Trainingsmenge durch das Netz zu propagieren (*Forward propagation*) und anschließend die Outputs mit den *ground-truth* Daten zu vergleichen. Eine Fehlerfunktion bestimmt daraufhin die Differenz zwischen den erwarteten und erhaltenen Daten und wird als Korrektur in das Netz gespeist. Diese Fehlerkorrektur wird dabei *Backpropagation* genannt und verläuft genau invers zur *Forward propagation* startend von der Outputschicht. Das *Backpropagation*-Verfahren passt dabei schichtweise die Gewichte zwischen den Neuronen an, um den Fehler zu minimieren. Um dabei aufgrund der Vielzahl von Neuronen und Gewichten eine effiziente Berechnung der Fehlerkorrektur zu gewährleisten, wird meist ein Gradientenabstiegsverfahren genutzt, welches während der Fehleroptimierung lokale Minima findet, anstatt alle möglichen Kombination von Gewichten zu berücksichtigen. Hierbei werden die Fehlerfunktion als mehrdimensionale Oberfläche (in Abhängigkeit der Gewichte) betrachtet und die Anpassung der Parameter/Gewichte entlang der Richtung des stärksten Abstiegs vorgenommen.

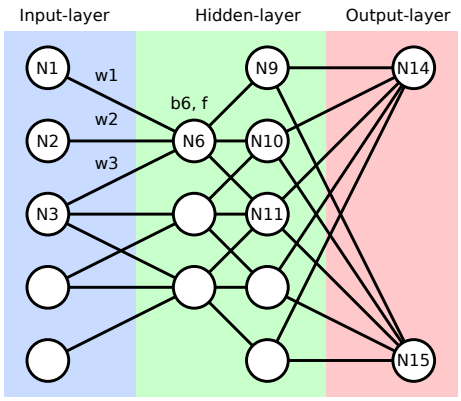
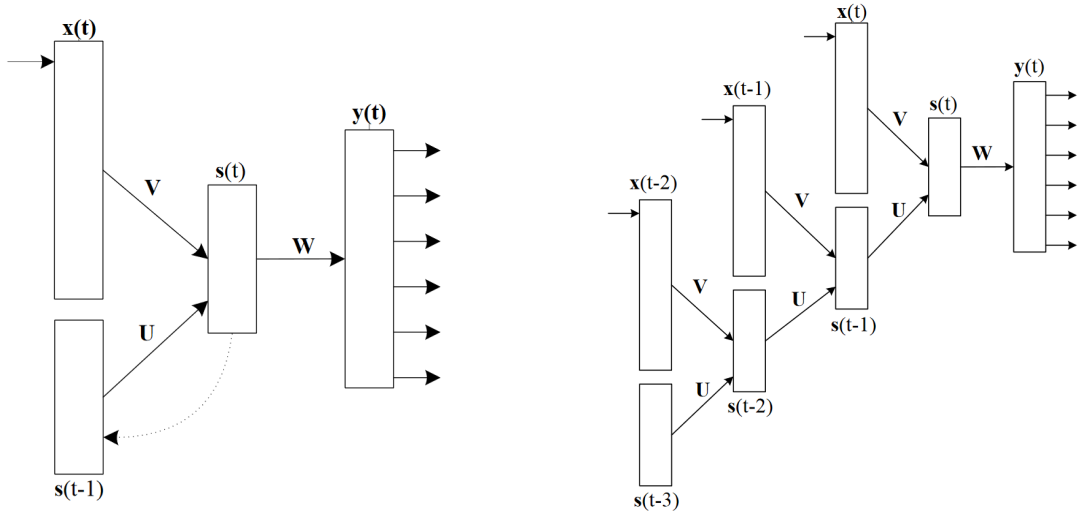


Abbildung 6: Vereinfachtes Beispiel eines neuronalen Netzes. Knoten repräsentieren Neuronen. Schichten sind farbig markiert. (Eigene Abbildung)

Eine besondere Art von neuronalen Netzen stellen *Recurrent neural networks* (*RNN*) dar, welche im Gegensatz zu klassischen *Feed forward*-Netzen auch Schleifen und Kreise in der Graphstruktur erlauben. Diese Rückkanten oder Schleifen ermöglichen diesen Netzen zeitlich aufeinander folgende Daten zu verarbeiten und dabei auf eventuelle zeitliche Abhängigkeiten dieser Daten zu reagieren. Das Training erfolgt hierbei analog zum Training von *Feed-forward*-Netzen durch *Backpropagation through time*, einer Variante der *Backpropagation* wobei das *RNN* zeitlich entfaltet wird (siehe Abbildung 7).

Ein großes Problem dieser Netze stellt die Fähigkeit zu Lernen mittels *Backpropagation through time* dar, da häufig die Intensität der Fehlerfunktion ähnlich zu sehr tiefen *Feed-forward*-Netzen über die Schichten nachlässt oder in selteneren Fällen immer stärker wird. Diese Probleme, welche unter den Namen *Vanishing gradient* bzw. *Exploding gradient* bekannt sind, werden durch die Anwendung von *Long-short-term memory* (*LSTM*) (Hochreiter und Schmidhuber, 1997) gelöst. Diese Variante von *RNN* nutzt zusätzliche *Gates*, welche ähnlich zu Speicherbausteinen aufgebaut sind und dafür sorgen, dass der für *Backpropagation* wichtige Gradient weder zu klein wird (Netz kann in den ersten Schichten nicht lernen: *Vanishing gradient*) noch zu groß wird (Fehler werden überkorrigiert: *Exploding gradient*). Aus diesem Grund bieten sich *LSTM*-Netze in Bereichen an, die eine hohe Komplexität benötigen, da diese Bereiche aufgrund der Anzahl an Schichten bzw. Rückkanten am stärksten durch das *Vanishing Gradient*-Problem betroffen sind. LSTMs können darüber hinaus erweitert werden, indem mehrere LSTM-Schichten mit Gates verwendet werden. Auf diese Weise wird das Netz tiefer und kann potenziell komplexere Zusammenhänge erkennen.

Eine noch speziellere Variante von *LSTM*-Netzen stellen *Bidirectional long-short-term memory*-Netze (*BLSTM*) dar. Diese basieren auf *Bidirectional recurrent neural networks* nach Schuster und Paliwal (1997) und unterscheiden sich von *RNN* dadurch, dass auch zeitlich zukünftige Informationen in die Berechnung der Neuronenaktivität eingehen können. Anschaulich bedeutet dies, dass ein wie in Abbildung 7 entfaltetes Netz nicht nur gerichtete Kanten von früheren Zuständen besitzt, sondern auch gerichtete Kanten zurück von späteren Zuständen. Ein Vorteil dieser Variation ist z.B. im Bereich der Textanalyse die Berücksichtigung des gesamten Kontextes, sowohl vor dem aktuellen Input als



(a) Beispiel für ein einfaches *RNN*

(b) Über drei Schichten entfaltetes *RNN*

Neural layer	Description	Index variable
$x(t)$	input layer	i
$s(t-1)$	previous hidden (state) layer	h
$s(t)$	hidden (state) layer	j
$y(t)$	output layer	k

Weight matrix	Description	Index variables
V	Input layer $\rightarrow$ Hidden layer	i, j
U	Previous hidden layer $\rightarrow$ Hidden layer	h, j
W	Hidden layer $\rightarrow$ Output layer	j, k

(c) Erklärungen der Bezeichnungen

Abbildung 7: Beispiel für die Entfaltung einer *RNN* für die Anwendung von *Backpropagation through time*. Entnommen aus Guo (2013).

auch nach dem aktuellen Input.

Eine weitere Art von neuronalen Netzen stellen Convolutional Neural Nets (CNN) dar. Diese Art von Netzen wird meist im Bereich der Bilderkennung verwendet, da sie anschaulich auf den sogenannten Convolutional Ebenen einen Filter auf das Bild anwendet und anschließend in einer Pooling Ebene eine Datenreduktion, z.B. durch Auswahl des aktivsten Neurons in einer zugehörigen Filtermatrix, auch Max-Pooling genannt, durchführt. Diese Art von Netzen kann ebenfalls effektiv für Textklassifikationsaufgaben eingesetzt werden (siehe Guggilla et al. (2016)).

Eine spezifische Art von Layer stellen Embedding-Layer dar, welche im neuronalen Netz automatisch Embeddings für den Input erzeugen. Diese Embedding-Layer können sowohl durch das Training des Netzes komplett ohne vorheriges Training aufgebaut oder auch mit vortrainierten Embeddings konstruiert werden.

In dieser Arbeit werden, analog zu Liebeck (2018) CNNs, LSTMs, BLSTMs sowie Stacked LSTMs (Goldberg, 2016) verwendet. Die im Speziellen verwendeten Verfahren sowie einige der wichtigsten Parameter sind in Tabelle 14 dargestellt. Im Weiteren werden die Verfahren mit den in der Spalte „Name“ angegebenen Abkürzungen (ggf. in Plots weiter abgekürzt z.B. LSTM - e statt LSTM - empty) bezeichnet.

4.4 Features

Im Folgenden werden die in dieser Arbeit verwendeten Repräsentationen (Features) der Textdaten für die Klassifikationsaufgaben beschrieben.

4.4.1 N-Gramme

N-Gramme stellen die Zerlegung eines Textes in alle Teiltexte der Länge N dar. Dies kann auf verschiedenen Granularitätsebenen erfolgen und somit beispielsweise auf Zeichenebene (Character-N-Gramms) alle Teilwörter der Länge N eines Wortes sein oder auf Wortebene alle Teilsätze der mit N Wörtern. Zusätzlich können auch Bereiche für N (z.B. 3-5) verwendet werden, was der Verwendung aller 3- bis 5-Grammen entspricht. In dieser Arbeit werden N-Gramme der Länge 4 (auch als 4-4 bezeichnet) sowie der Kombination von N-Grammen der Längen 3-5 verwendet und dasjenige N mit den besten Ergebnissen während der Parameterfindung zur finalen Evaluation ausgewählt.

Die Repräsentation der N-Gramme erfolgt schließlich analog zu einem Bag of Words Modell.

4.4.2 Bag of Words

Das Bag of Words Modell stellt eine einfache Repräsentation von Texten zu einem gegebenen Vokabular dar. Dazu wird meist ein Vokabular der Länge M aus allen im Korpus vorkommenden Wörtern gebildet und jedes einzelne Dokument durch einen Vektor der Länge M repräsentiert. Diese Vektoren geben für jedes Wort des Vokabulars die Anzahl an Vorkommen im jeweiligen Dokument an. Ein Problem dieser einfachen Methode stellt

Name	Beschreibung	Komponenten	Parameter
BLSTM	Bidirectional LSTM	Embedding LSTM (bidirektional) Dense	Dim: 300 Dim: 128, Drop: 0.2
E-CNN	CNN mit Embedding Layer	Embedding Convolution Layer Max-Pooling Dense	Dim: 300 Dim: 175, Kernel: 3 Drop: 0.2
E-CNN-LSTM	E-CNN konkateniert mit LSTM	Embedding Dropout Convolution Layer Max-Pooling LSTM Dense	Dim: 300 Wert: 0.7 Dim: 100, Kernel: 5 Dim: 70
E-LSTM-empty	LSTM mit Embedding Layer	Embedding LSTM Dense	Dim: 300 Dim: 128, Drop: 0.2
E-LSTM-pre	LSTM mit trainiertem Embedding Layer	Embedding LSTM Dense	Dim: 300 Dim: 128, Drop: 0.9
LSTM - stacked	2 Konkatenierte LSTM	Embedding LSTM LSTM Dense	Dim: 300 Dim: 128, Drop: 0.7, Recurent-Drop: 0.5 Dim: 64, Drop: 0.7, Recurent-Drop: 0.5

Tabelle 14: Übersicht über die verwendeten Deep Learning Klassifikatoren mit ausgewählten Parametern. Alle Verfahren verwenden 10 Trainingsepochen. Die Wahl der Parameter erfolgte durch ein Gridsearch. Für vortrainierte Embeddings wurden Word2Vec Embeddings, welche auf einem deutschen Wikipedia-Korpus trainiert wurden, verwendet.

die gleichmäßige Gewichtung aller Wörter dar. Häufig vorkommende Wörter (Stoppwörter) besitzen einen großen Einfluss auf die Repräsentation eines Textes, während seltene, semantisch wichtige Wörter durch ihre geringe Frequenz keinen außergewöhnlichen Einfluss auf die Repräsentation besitzen. Auch die Länge der Dokumente ändert die Repräsentationen durch die meist höheren Frequenzen maßgeblich. Um dies zu beheben können TF-IDF gewichtete Repräsentationen verwendet werden. Diese gewichten jedes Wort (Term) abhängig von ihrem Vorkommen in einem Dokument und der Dokumentlänge und normieren dieses Gewicht invers zum Vorkommen des Terms in allen Dokumenten des Korpus. Auf diese Weise können seltene, prägnante Terme sowie kürzere Dokumente stärker gewichtet werden. Formal lassen sich ein TF-IDF Gewicht für ein Dokument $d \in D$ aus der Menge D von N Dokumenten und Term t wie folgt beschreiben:

$$\begin{aligned} tf(t,d) &= \frac{f_{t,d}}{\max\{f_{t',d} : t' \in d\}} \\ idf(t,D) &= \log \frac{N}{|\{d \in D : t \in d\}|} \\ tf\text{-}idf(t,d,D) &= tf(t,d) \cdot idf(t,D) \end{aligned} \tag{25}$$

, wobei $f_{t,d}$ die Frequenz eines Terms in einem Dokument bezeichnet.

4.4.3 Grammatikalische Features

In dieser Arbeit werden auch grammatikalische Features verwendet. Dazu werden durch spaCy bestimmte Part-of-Speech Tags für jedes Token genutzt, um für die zu klassifizierende Einheit ein Histogramm über die POS-Tags zu generieren. Zusätzlich dazu werden Histogramme ebenfalls über Dependencies, d.h. Abhängigkeiten der Wörter untereinander, erstellt. Hierzu werden ebenfalls durch spaCy generierte Dependencies genutzt.

4.4.4 Word Embeddings

Als Repräsentationen der einzelnen Tokens werden in dieser Arbeit als weiteres Feature Word-Embeddings verwendet. Word Embeddings stellen eine Art der Abbildung von Wörtern auf reellwertige Vektoren dar, welche anschaulich eine semantische Ähnlichkeit zueinander besitzen sollen, wenn sie im Vektorraum nahe beieinander liegen. Eine Technik zur Erzeugung von Word Embeddings stellt das word2vec Verfahren von Mikolov et al. (2013a) dar (siehe auch Goldberg und Levy (2014)). Dieses Verfahren nutzt das Training eines neuronalen Netzes um in dessen Struktur semantisch wertvolle Informationen zu speichern (siehe Abbildung 8). Aufgabe eines solchen Netzes ist die Bestimmung der Wahrscheinlichkeiten eines jeden Wortes im Vokabular, dass sie sich in der Nähe, d.h. innerhalb eines k-Wörter großem Fensters, zu einem bestimmten Wort befinden. Im Gegensatz zu üblichen Anwendungen von neuronalen Netzen wird hierbei die eigentliche Aufgabe des Netzes im späteren Verlauf nicht mehr benötigt. Ziel ist es nur durch die benutze Trainingsaufgabe ein Training der Gewichte in den Hidden-Layern des Netzes zu erreichen. Ist ein solches Netz nun für jedes Wort des Vokabulars trainiert, werden die in den Hidden-Layern gespeicherten Gewichte zur Repräsentation der Wörter verwendet.

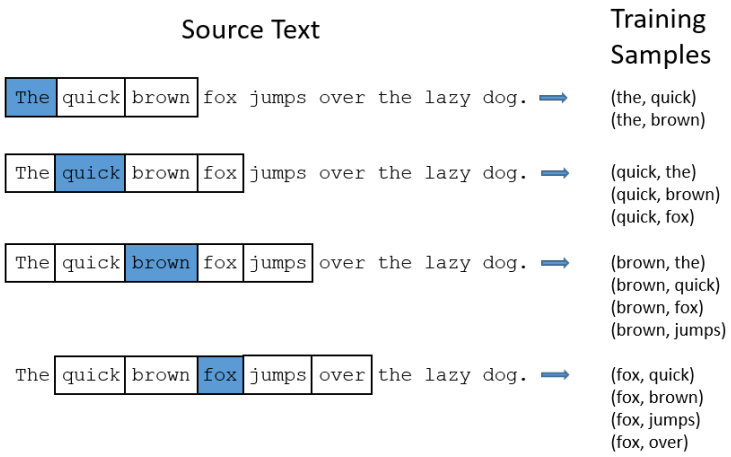


Abbildung 8: Trainingsdaten für ein word2vec Modell. Im Beispiel wird eine Fenstergröße von $k = 2$ und der Satz „The quick brown fox jumps over the lazy dog.“ verwendet. Entnommen aus McCormick (2016)

Die Anzahl der Neuronen in den Hidden-Layern gibt dabei die Größe der resultierenden Feature-Vektoren vor. Eine Veranschaulichung des Netzes aus McCormick (2016) findet sich in Abbildung 9. Ursprünglich wurden von Google dabei 300-dimensionale Featurevektoren gebildet. Eine Übersicht über das Verfahren und Optimierungen findet sich in McCormick (2016) und McCormick (2017).

In dieser Arbeit werden 100-, 200- und 300-dimensionale, auf einem Wikipedia-Korpus trainierte, word2vec Embeddings verwendet.

4.4.5 Character Embeddings

Character Embeddings können analog zu Word Embeddings mit der Anwendung auf N-Grammen verstanden werden. Hierbei werden für ein gegebenes Wort erst alle N-Gramme für ein N oder einen Bereich von Werten von N erstellt und dann Repräsentationen für Wörter durch Summierung aller analog zu Kapitel 4.4.4 entstehenden N-Gramm-Repräsentationen des Wortes gebildet. Ein bekanntes Verfahren, welches auch in dieser Arbeit eingesetzt wird, stellt das von Facebook entwickelte fastText-Verfahren (Bojanowski et al., 2017,Joulin et al., 2017) dar. Ein Vorteil dieses Verfahrens gegenüber Word Embeddings ist unter anderem die Möglichkeit, potenziell semantisch sinnvolle Embeddings auch für Wörter außerhalb des Vokabulars erstellen zu können, da diese meist N-Gramme enthalten, die im Vokabular ebenfalls vorkommen. Außerdem werden zum Training von selten vorkommenden Wörtern meist auch häufig vorkommende N-Gramme genutzt, wodurch das Training effektiver werden kann. In dieser Arbeit werden 100-dimensionale fastText Embeddings mit N-Gramm-Größen 3-6 und 4-4, welche auf einem deutschen Wikipedia-Korpus trainiert wurden, sowie 300-dimensionale fastText Embeddings mit den performantesten N-Grammgrößen zwischen 3-6 und 5-5, welche auf dem Common Crawl Korpus trainiert wurden, verwendet (siehe Grave et al. (2018)).

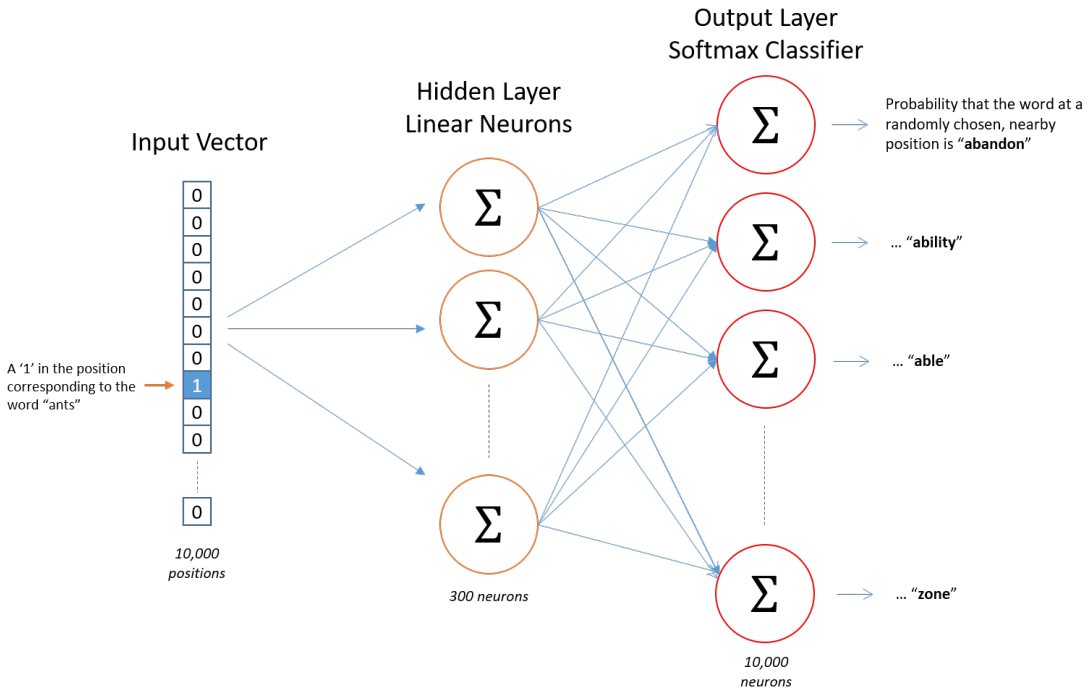


Abbildung 9: Veranschaulichung des verwendeten neuronalen Netzes zur Erstellung von Word2Vec Repräsentationen. Das Vokabular besteht aus 10000 Wörtern und die resultierenden Repräsentationen sollen 300 Dimensionen besitzen. Im Beispiel wird das Wort „Ants"durch einen one-hot Vektor der Länge 10000 repräsentiert. Entnommen aus McCormick (2016)

4.4.6 Latent Dirichlet Allocation

Ein weiteres in dieser Arbeit verwendetes Feature stellt die Latent Dirichlet Allocation (LDA) nach Blei et al. (2002) und Blei et al. (2003) dar. Dieses generative statistische Verfahren modelliert die Verteilung von Themen zu gegebenen Dokumenten und die Wahrscheinlichkeiten von Themen, spezifische Wörter zu generieren. Angelehnt an Chen (2018b) folgt eine anschauliche Erklärung des Verfahrens:

Seien beispielsweise die Sätze (Dokumente) aus Tabelle 15 gegeben, so können durch Anwendung von LDA die Wahrscheinlichkeiten berechnet werden, inwieweit die einzelnen Sätze zu einer vorher festgelegten Anzahl K an Themen passen. Dies könnten hier für $K = 2$ folgende Wahrscheinlichkeiten sein:

Sätze 1 und 2: 100% Thema A.

Sätze 3 und 4: 100% Thema B.

Satz 5: 60% Thema A und 40% Thema B.

Mit den Themen (Verteilungen nur Auszugsweise über die wahrscheinlichsten Wörter dargestellt):

Satz Nr.	Satz
1	Ich esse gerne Brokkoli und Bananen.
2	Ich esse eine Banane und einen Spinatsmoothie zum Frühstück.
3	Welpen und Kätzchen sind süß.
4	Meine Schwester hat gestern ein Kätzchen adoptiert.
5	Schau dir diesen süßen Hamster an, wie er am Brokkoli knabbert.

Tabelle 15: Beispielsätze für die Veranschaulichung der Latent Dirichlet Allocation. Angelehnt an Chen (2018b).

Thema A: 30% Brokkoli, 15% Bananen, 10% Frühstück, 10% knabbert, ... (Dieses Thema repräsentiert Essen)

Thema B: 20% Welpen, 20% Kätzchen, 20% süß, 15% Hamster, ... (Dieses Thema repräsentiert somit süße Tiere)

Die Funktionsweise der Latent Dirichlet Allocation basiert dabei auf der Annahme, dass Dokumente durch einen spezifischen generativen Prozess entstehen. Für die Erstellung eines Dokumentes wird zuerst die Anzahl der Wörter N als Zufallszahl einer Verteilung (im Speziellen einer Poisson-Verteilung) ermittelt. Darauf wird die Themenverteilung des Dokumentes durch eine Dirichlet-Verteilung über K Themen ermittelt. Für jedes zu generierende Wort in einem Dokument wird nun, ausgehend von der zuvor ermittelten Themenverteilung des Dokumentes, ein Thema ausgewählt. Nun wird abhängig von einer ebenfalls anfangs zufälligen Themen-Wort-Verteilung ein Wort ausgehend von dem gewählten Thema generiert.

Für den Einsatz von LDA zur automatisierten Themenfindung wird ein iterativer Prozess zur Erstellung der Wort-Thema-Verteilungen und Dokument-Thema-Verteilungen verwendet. Die Ergebnisse dieses Prozesses sind die inferierten Themen, Themen-Wort-Verteilung und Dokument-Themen-Verteilung, welche für die Anwendung als Feature in Textklassifikationsaufgaben interessant sind. Der dafür notwendige iterative Prozess, welcher auch als Gibbs Sampling bezeichnet wird, kann für eine vom Nutzer gewählte Themenanzahl K wie folgt beschrieben werden:

Zuerst werden alle Wörter der Dokumente zufällig je einem der K Themen zugeordnet. Auf diese Weise entstehen, wenn auch meist schlechte, Themen-Dokument aber auch Wort-Thema Verteilungen. Um diese nun iterativ zu verbessern, werden für jedes Dokument d, Wort $w \in d$ und Thema t zwei Maße $p(t|d)$ und $p(w|t)$ berechnet. Der Wert $p(t|d)$ gibt dabei den Anteil der Wörter in d an, welche dem Thema t zugeordnet sind, und $p(w|t)$ gibt den Anteil der Zuordnungen von Wort w zu Thema t über alle Dokumente an. Nun wird die Themenzuordnung von Wort w durch Thema t, welches mit Wahrscheinlichkeit $p(t|d) \cdot p(w|t)$ gezogen wird, aktualisiert. Dies entspricht anschaulich der Wahrscheinlichkeit im generativen Modell, dass Thema t Wort w erzeugt hat. Somit wird in diesem Schritt die Themenzuordnung eines Wortes, ausgehend von allen anderen Zuordnungen, durchgeführt. Dieser Schritt wird solange wiederholt, bis die Zuordnungen von Wörtern zu Themen stabil bleibt bzw. eine maximale Anzahl an Iterationen erreicht wurde. Auf diese Weise können sehr zutreffende Wort-Themen- und Themen-Dokument-Verteilungen gefunden werden.

Die hier vorgenommene Beschreibung der Latent Dirichlet Allocation, angelehnt an Chen (2018b), stellt dabei ein starke Vereinfachung dar. Für genauere Informationen wird auf Blei et al. (2003) verwiesen.

In dieser Arbeit werden mehrere LDA-Modelle verwendet: Ein auf Wikipedia trainiertes Modell mit $K = 300$ sowie auf den verwendeten Datensätzen trainierte LDA-Modelle mit $K = 15$ und $K = 25$ für THF und Braunkohle sowie $K = 15$, $K = 25$ und $K = 50$ für den kombinierten Korpus.

4.5 Granularitätsebenen

Neben der Auswertung der von Liebeck (2018) verwendeten Klassifikatoren und Features auf zwei neuen Korpora sowie einem vereinten Korpus stellt die Untersuchung der Klassifikationsaufgaben auf verschiedenen Granularitätsebenen einen Kernbestandteil dieser Arbeit dar. Hierbei werden die drei in Kapitel 1.2.4 vorgestellten Granularitätsebenen verwendet, um die anstehenden Klassifikationen auf Tokenebene, Satzebene sowie Dokumentebene durchzuführen. Hierbei wird abweichend von Habernal und Gurevych (2017) auf der feineren Granularitätsebene jedes Token einzeln klassifiziert.

Die Verfahrensweisen zur Adaption des von Liebeck (2018) entwickelten Frameworks für die weiteren Granularitätsebenen werden im Folgenden vorgestellt.

4.5.1 Sentence Labeling

Die Klassifikation von Sätzen wurde bereits von Liebeck (2018) durchgeführt, sodass mit nur wenigen Änderungen bezüglich des Eingabeformates eine vollständige Kompatibilität zwischen den verwendeten Frameworkkomponenten und der Klassifikation der neuen Korpora auf Satzebene gewährleistet werden konnte. Inhaltlich wird dabei jeder Satz als einzelnes Dokument aufgefasst und klassifiziert.

4.5.2 Document Labeling

Die Abstraktion der Klassifikationsaufgaben auf Dokumentenebene lässt sich mit dem vorliegenden Framework bereits durch eine Änderung des Eingabeformates lösen. Werden anstatt von Sätzen ganze Beiträge als Konkatenation der in ihnen enthaltenen Sätze als Dokumente aufgefasst, können die von Liebeck (2018) implementierten Features und Klassifikatoren ebenfalls verwendet werden.

4.5.3 Sequence Labeling

Für die Klassifikation auf Tokenebene (Sequence Labeling) musste eine neue Art der Datenrepräsentation gefunden werden. Besonders für die Verwendung der klassischen Machine-Learning Algorithmen (siehe Kapitel 4.2) ist es von Interesse, kontextreichen Features zu erzeugen, da einzelne Tokens selbst mit grammatischen Features häufig keine unterschiedlichen Repräsentationen erhalten, obwohl sie in verschiedenen Kontexten eingesetzt werden.

Diese Arbeit verwendet aufgrund dieses Problems folgenden Lösungsansatz: Für jedes Token wird ein künstlicher Satz durch die Konkatenation aller Tokens in einem Kontextfenster mit gegebener Länge m aus einem Dokument (hier: Beitrag) erzeugt. Diese Sätze können dann analog zum Sentence Labeling klassifiziert werden. Hierbei stellt das mittlere Token immer das eigentlich zu klassifizierende Token dar. Durch Variation der Fenstergröße (in der Anzahl von verwendeten Token) kann der Einfluss des Kontextes gesteuert werden.

Für Tokens, die sich am Anfang oder Ende eines Dokumentes befinden, werden die nicht vorhandenen Tokenplätze im Fenster nach einer von mehreren Padding-Methoden aufgefüllt. Implementiert wurden Zero-Padding, Document-Padding, welches Token aus einem benachbarten Dokument verwendet (d.h., falls im gleichen Dokument kein benachbarter Satz verfügbar ist), oder Mirror-Padding, welches die bestehende Tokenfolge gespiegelt anhängt bzw. voranstellt. Sind keinerlei benachbarte Token vorhanden, wird auf Zero-Padding, welches leere Tokens (Leerzeichen) verwendet, ausgewichen. In dieser Arbeit wird dabei nur die Zero-Padding Methode verwendet. Weitere Untersuchungen mit verschiedenen Padding-Methoden bleiben Forschung für zukünftige Arbeiten.

Ein Beispiel für die Erzeugung von Sätzen für Sequence Labeling Aufgaben findet sich in Abbildung 10.

Ursprünglicher Satz:

Baseball und Softball sind ein Freizeiterlebnis mit großem Unterhaltungsfaktor.

Generierte Sätze für Sequence Labeling:

##Baseball und Softball
#Baseball und Softball sind
Baseball und Softball sind ein
und Softball sind ein Freizeiterlebnis
Softball sind ein Freizeiterlebnis mit
sind ein Freizeiterlebnis mit großem
ein Freizeiterlebnis mit großem Unterhaltungsfaktor
Freizeiterlebnis mit großem Unterhaltungsfaktor.
mit großem Unterhaltungsfaktor.#
großem Unterhaltungsfaktor.##

Abbildung 10: Veranschaulichung der Erzeugung von Sätzen für Sequence Labeling Aufgaben. Blau markiert ist jeweils das originale Token, zu welchem ein Satz gebildet wird. Rot markierte Rauten bezeichnen das Padding-Token.

Aufgrund der ursprünglichen Implementierung der Klassifikatoren und Features für die Klassifikationen von Sätzen findet dabei keine Gewichtung der einzelnen Tokens in den so erzeugten künstlichen Sätzen statt. Für eine stärkere Gewichtung des zu klassifizierenden Tokens wird daher eine Vervielfachung dieses Tokens um einen Faktor k in der Tokenfolge vorgeschlagen. In dieser Arbeit werden auf diese Weise Ergebnisse für $k = 1$ (keine Vervielfachung) sowie $k = 2$ und $k = 3$ untersucht.

4.5.4 Intergranularität

In dieser Arbeit werden zusätzlich zu den Untersuchungen auf den drei zuvor beschriebenen Granularitätsebenen, Experimente mit gemischten Granularitäten durchgeführt, was im Folgenden als Intergranularität bezeichnet wird. Hierbei wird untersucht, ob das Training bzw. die Klassifikation auf einer feineren Granularitätsebene für eine Klassifikationsaufgabe auf gröberer Granularitätsebene genutzt werden kann. Dazu werden zwei Methoden untersucht.

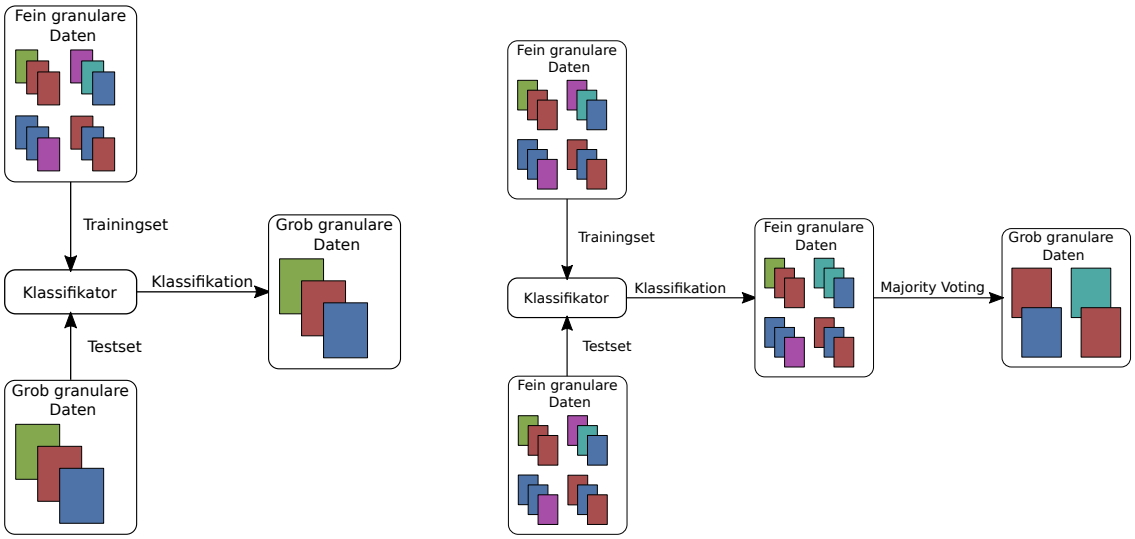
Eine erste Variante, die fortan als Klassifikation mit intergranularem Training bezeichnet wird, trainiert einen Klassifikator mit feingranularen Daten mit der Absicht den trainierten Klassfikator dann zur Klassifikation von grob granularen Daten zu nutzen. Hierbei müssen gegebenenfalls Labels, die aufgrund der verwendeten Granularität angepasst wurden (z.B. durch ein BIO-Schema im Fall des Sequence Taggings) vor dem Training zu ihren normalen Labels (ohne zusätzliches Schema) konvertiert werden. Nun kann ein Training auf dem feingranularen Datensatz mit den passenden Labels für die grobgranulare Klassifikation vorgenommen werden. Motiviert wird dieses Verfahren durch die Möglichkeit mehr Trainingsinstanzen durch eine feinere Granularität zu verwenden, sodass Klassifikatoren, deren Leistung stark von der Anzahl der Trainingsinstanzen abhängt, möglicherweise bessere Resultate liefern können als durch die Verwendung weniger, jedoch semantisch aussagekräftiger Trainingsinstanzen im Fall der groben Granularität. Ein Vorteil dieses Verfahren wäre zudem das Einsparen eines Trainings des Klassifikators auf grober Granularität, falls bereits die Absicht besteht mehrere Granularitäten zu untersuchen.

Eine zweite Variante wird durch die Verwendung von bereits durchgeführten Klassifikationen auf feiner Granularität motiviert. Hierbei werden die Ergebnisse eines vollständig feingranularen Trainings- und Klassifikationsprozesses in Ergebnisse einer grobgranularen Klassifikationsaufgabe konvertiert. Hierzu werden die entsprechend klassifizierten feingranularen Instanzen mit denjenigen grobgranularen Instanzen assoziiert, in welchen sie enthalten sind (so werden z.B. alle Sätze eines Dokumentes mit dem Dokument assoziiert). Daraufhin erfolgt die Vergabe des Labels für die assoziierte grobgranularen Instanzen per Majority Voting über die feingranularen Instanzen. Dieses Verfahren könnte inhaltlich ebenfalls die größere Anzahl an Trainingsinstanzen nutzen, um die Klassifikationsgenauigkeit zu erhöhen. Im Folgenden wird diese Methode als Klassifikation mit intergranularem Post Processing bezeichnet.

Eine Übersicht über beide Verfahren findet sich in Abbildung 11

4.6 Oversampling

Aufgrund der semantischen Diversität der während des Annotationsprozesess verwendeten Labels (siehe Kapitel 2.2.1) sind nicht alle Klassen mit gleicher Häufigkeit in den Datensätzen vorzufinden. Aufgrund dieser ungleichen Klassenverteilung existieren für einige Klassifikationsaufgaben (insbesondere Klassifikationsaufgaben C und E) wenige Datensätze, auf denen das Training durchgeführt werden kann. In dieser Arbeit wird aus diesem Grund der Einfluss von Oversampling-Algorithmen auf die Klassifikationsergebnisse exemplarisch untersucht. Oversampling bezeichnet dabei die Erzeugung zu-



(a) Klassifikation mit intergranularem Training

(b) Klassifikation mit intergranularem Post Processing

Abbildung 11: Veranschaulichung der verwendeten Intergranularitätmethoden.

sätzlicher Datenpunkte, wobei im Speziellen unterrepräsentierte Klassen zusätzliche Datenpunkte erhalten sollen. In dieser Arbeit werden die beiden durch imbalanced-learn-Bibliothek[17] (Lemaître et al., 2017) implementierte Verfahren SMOTE (Chawla et al., 2002) und ADASYN (He et al., 2008) verwendet, welche im Gegensatz zu Random Oversampling, welches existierende Datenpunkte dupliziert, neue Datenpunkte durch Interpolation erzeugen. Es folgt eine an die Erklärungen von imbalanced-learn[18] angelehnte Erläuterung beider Verfahren.

4.6.1 SMOTE (Synthetic minority oversampling technique)

SMOTE (Chawla et al., 2002) generiert einen zusätzlichen Datenpunkt x_{neu} für einen Datenpunkt x_i einer Minderheitsklasse, indem x_{neu} zwischen x_i und einem Datenpunkt x_{zi} aus der k-Nachbarschaft von x_i interpoliert wird:

$$x_{\text{neu}} = x_i + \lambda \cdot (x_{zi} - x_i) \text{ mit } \lambda \in [0, 1] \tag{26}$$

Die Wahl der verwendeten Datenpunkte x_i und x_{zi} unterscheidet sich dabei abhängig von der verwendeten Variante von SMOTE. Die reguläre SMOTE Variante schränkt die Wahl von x_i und den daraus resultierenden x_{zi} in seiner k-Nachbarschaft nicht ein. Die „borderline“ Variante von SMOTE klassifiziert hingegen zuerst jeden Datenpunkt x_i durch Betrachtung seiner k Nachbarn in eine von drei Kategorien: Gehören alle Nachbarn von x_i einer anderen Klasse als x_i an, so wird x_i als „Rauschen“ klassifiziert. Gehören mehr als die Hälfte der Nachbarn einer anderen Klasse als x_i an, so wird x_i als

[17] https://imbalanced-learn.readthedocs.io/en/stable/
[18] https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html

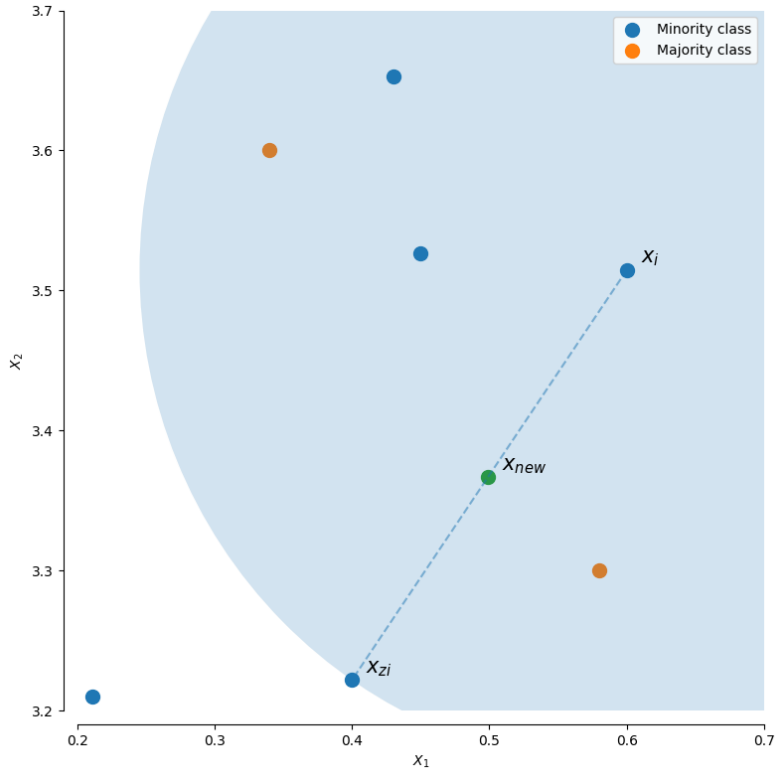


Abbildung 12: Veranschaulichung der Erzeugung neuer Datenpunkte durch SMOTE. Dabei bezeichnet x_i einen gegebenen Datenpunkt und x_{zi} stellt einen Datenpunkt aus der k-Nachbarschaft von x_i dar. Entnommen der Website von imbalanced-learn[19].

„gefährdet“ klassifiziert und falls weniger als die Hälfte der Nachbarn eine andere Klassenzugehörigkeit aufweisen, erfolgt eine Klassifizierung als „sicher“. Die beiden SMOTE Varianten Boderline-1 und -2 wählen nun nur diejenigen x_i als Kandidaten zur neuen Datenpunkgenerierung aus, welche als „gefährdet“ klassifiziert wurden. Im Falle von Borderline-1 muss x_{zi} zur gleichen Klasse wie x_i gehören, während Borderline-2 diese Einschränkung nicht besitzt. Eine weitere Variante von SMOTE nutzt eine vorläufig trainierte SVM zur Identifikation von Grenzen zwischen den einzelnen Klassen und erzeugt daraufhin neue Datenpunkte vermehrt in der Nähe dieser Grenzen (Nguyen et al., 2011).

4.6.2 ADASYN (Adaptive Synthetic Sampling Approach)

ADASYN (He et al., 2008) kann als Modifikation von SMOTE verstanden werden und generiert neue Datenpunkte analog zu SMOTE, passt dabei allerdings die Anzahl der generierten Punkte an die Verteilung der Klassen in der Nachbarschaft von x_i an. Je mehr Datenpunkte in der k-Nachbarschaft von x_i einer anderen Klasse als x_i angehören, desto mehr neue Datenpunkte werden dort generiert.

5 Evaluation

In diesem Kapitel werden die Ergebnisse der in den Kapiteln 2 und 4 erläuterten Untersuchungen vorgestellt. Dazu werden in Kapitel 5.1 die Sentence-Tagging Ergebnisse auf mittlerer Granularitätsebene vorgestellt. In Kapitel 5.2 werden die Sequence-Tagging Ergebnisse auf feiner Granularitätsebene und in Kapitel 5.3 die Document-Tagging Ergebnisse auf grober Granularitätsebene präsentiert (siehe auch Kapitel 1.2.4 für eine Erläuterung der Granularitätsebenen). In Kapitel 5.4 folgen die Ergebnisse der Intergranularitätsuntersuchungen (vergleiche auch Kapitel 1.2.4 und 4.5.4). Jedes Kapitel der Ergebnisse auf verschiedenen Granularitätsebenen ist dabei zusätzlich nach Verfahrensweise (Klassisches Machine Learning bzw. Deep Learning (siehe Kapitel 4)) und Klassifikator gegliedert. Aufgrund der Anzahl und Größe der detaillierten Ergebnistabellen wurden diese in den Anhang (Kapitel A) verschoben. Zusammenfassende Abbildungen und Tabellen finden sich neben der Analyse und Erläuterungen in den folgenden Kapiteln. Gestrichene Einträge in den Ergebnistabellen konnten aufgrund von Laufzeitüberschreitungen im Jobsystem der verwendeten HPC-Servers nicht bis zum Ende ausgeführt werden und konnten aus zeitlichen Gründen nicht wiederholt werden.

5.1 Sentence-Tagging

In diesem Kapitel werden Die Ergebnisse des Sentence-Labelings auf mittlerer Granularitätsebene vorgestellt. Alle detaillierten Ergebnisse dazu befinden sich in Kapitel A.1.

5.1.1 Klassisches Machine Learning

Es folgen die Ergebnisse für die Klassifikatoren Random Forest (RF), Support Vector Machine (SVM), Support Vector Machine mit linearem Kernel (SVM-Linear) sowie Conditional Random Fields (CRF). Detaillierte Informationen hierzu sind im Anhang in Kapitel A.1.1 zu finden.

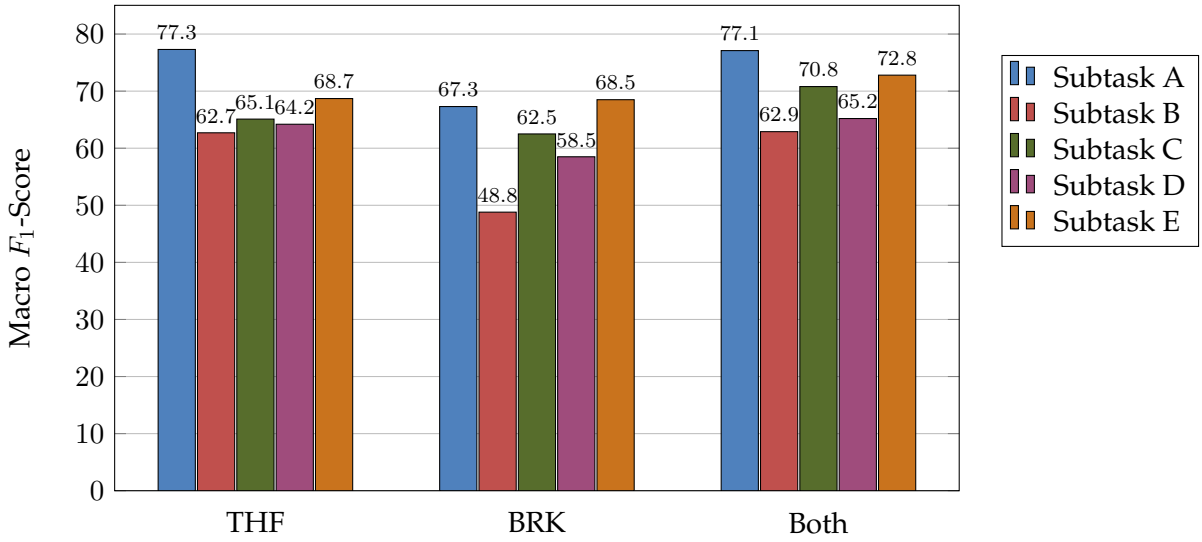


Abbildung 13: Beste Sentence Tagging Ergebnisse für den **Klassifikator Random Forest**.

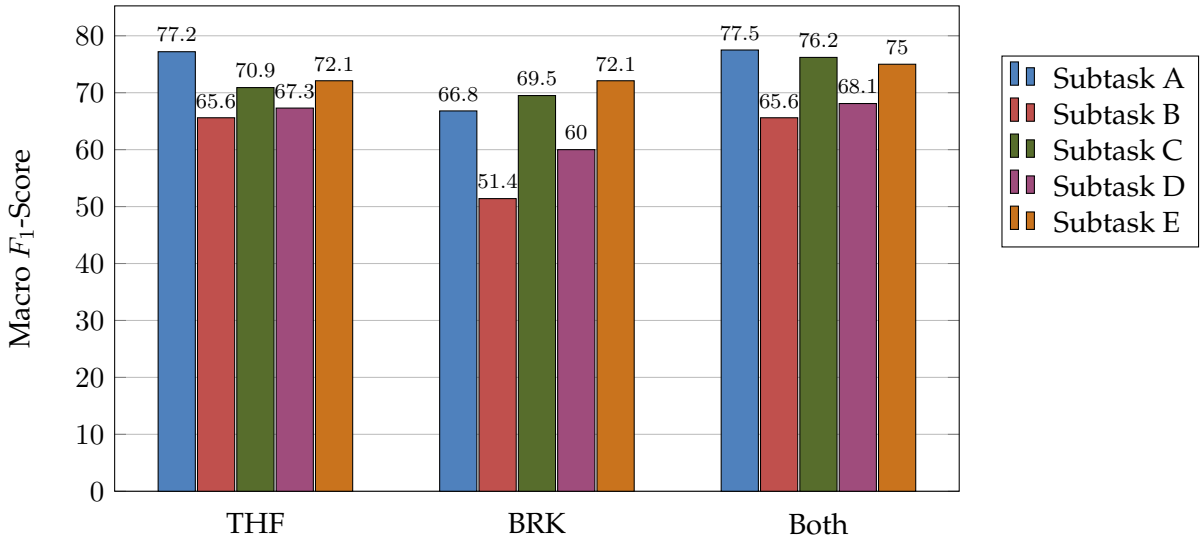


Abbildung 14: Beste Sentence Tagging Ergebnisse für den **Klassifikator Support Vector Machine**.

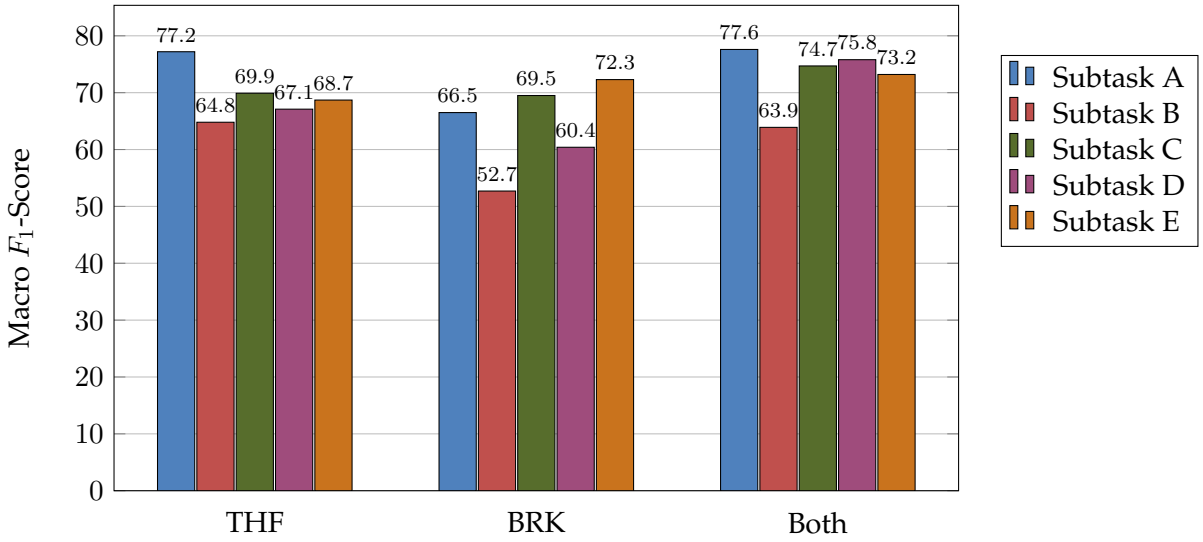


Abbildung 15: Beste Sentence Tagging Ergebnisse für den **Klassifikator Support Vector Machine: Linearer Kernel**.

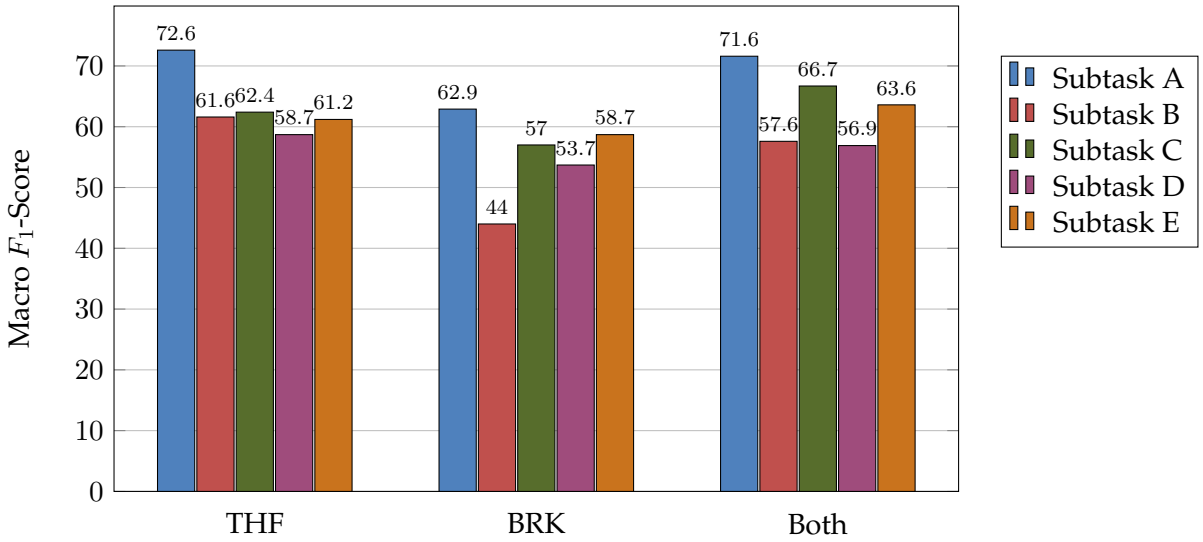


Abbildung 16: Beste Sentence Tagging Ergebnisse für den **Klassifikator CRF**.

Bezüglich der Subtasks zeigt sich ein klares Bild: Die binäre Klassifikationsaufgabe Subtask A (Erkennung von argumentativen Sätzen) erzielte mit allen Verfahren die besten Ergebnisse. Mit einem über die besten Ergebnisse aller Verfahren gemittelten Macro F_1-Score von 76.1% konnten die Verfahren diese Aufgabe am besten lösen. Die Erkennung von argumentativen Sätzen scheint somit unter den Subtasks die einfachste Aufgabe darzustellen. Aus den Erfahrungen der manuellen Codierung entspricht das auch der menschlichen Annahme.

Deutlich schwächer fallen die Ergebnisse für Subtask B aus. Die Erkennung der drei Argumentationskomponenten Vorschlag, Argument und Positionierung fällt hier merklich schlechter aus. Dies kann sowohl durch die größere Anzahl an Klassen als auch durch die Schwierigkeit der Aufgabe selbst begründet sein. Schon während der manuellen Codierung war es häufig schwierig, Argumente und Vorschläge bzw. Argumente und Positionierungen voneinander zu trennen. Daher entspricht dieses Ergebnis den eigenen Erwartungen.

Subtask C (Unterscheidung zwischen positiven und negativen Positionierungen) schneidet wiederum als binäres Klassifikationsproblem deutlich besser als Subtask B jedoch schlechter als Subtask A ab. Dies kann mit der Schwierigkeit, die Seite der Positionierung zu erkennen, begründet werden, was im Vergleich zur Erkennung von rein argumentativen Sätzen schwieriger fällt. Besonders der Einfluss von Verneinungen bzw. Invertierungen kann hier leicht zu falschen Ergebnissen führen, während Subtask A durch quantitativere Betrachtungen (z.B. Anzahl der Adjektive, Adverbien etc.) robuster ausfallen konnte. Zusätzlich stehen für Subtask C deutlich weniger Trainingsinstanzen als für Subtask A zur Verfügung, was die Klassifikationsgüte weiter beeinflussen kann.

Subtask D (Erkennung von Emotionen) zeigt eine Ungewöhnlichkeit. Fast alle Klassifikatoren weisen schlechtere Ergebnisse als Subtask A auf. Nur die Support Vector Machine mit linearem Kernel kann hier auf dem kombinierten Datensatz verhältnismäßig gut abschneiden. Es scheint als ob dieses Verfahren mit steigender Korpusgröße im Vergleich zu den anderen Verfahren besser skaliert. Dabei muss allerdings angemerkt werden, dass die SVM mit linearem Kernel auch den höchsten durchschnittlichen Macro F_1-Score von allen Verfahren aufweist (siehe Abbildung 17 und Tabelle 20 sowie Tabelle 53 im Anhang). Diese Beobachtung wiederholt sich auch später in den Ergebnissen des Document Taggings in Kapitel 5.3.1 und Sequence Taggings in Kapitel 5.2.2.

Subtask E schneidet ähnlich zu Subtask C ab. Die Unterscheidung zwischen positiven und negativen Emotionen kann ebenfalls von wenigen Wörtern (Verneinungen etc.) abhängen.

Hinsichtlich der Klassifikatoren zeigen sich SVM und SVM-linear dominant. Beide Verfahren schneiden durchschnittlich am besten ab. Wird die Fähigkeit auch Minderheitsklassen zu lernen betrachtet, zeigen diese sowie der Random Forest Klassifikator mit Macro F_1-Scores von 62.2% (SVM), 62.5% (SVM-LIN) und 61.2% (RF) die besten Ergebnisse während der Conditional Random Fields Klassifikator mit 56.8% deutlich schlechter abschneidet. Wird hingegen der Micro F_1-Score betrachtet, welcher die untervertretenen Klassen schwächer gewichtet, so schneidet der im Macro F_1-Score deutlich zurückliegende Klassifikator Conditional Random Fields mit 71.4% verhältnismäßig gut ab, was vermuten lässt, dass die CRF zu einer fälschlichen Zuordnung zur Mehrheits-

Klasse	Precision	Recall	F_1-Score	Support
MajorPosition	49.8	23.5	31.9	592
Claim	23.1	5.1	8.4	175
Premise	87.4	96.4	91.7	4357
Summe	80.8	84.8	81.9	5124

Tabelle 16: Ergebnisse pro Klasse. Klassifikator: CRF, Subtask: B, Feature: LDA Wikipedia, Datensatz: BRK. Macro F_1-Score: 44.0

Klassifiziert / Gold	Claim	MajorPosition	Premise	Summe
Claim	43	29	103	175
MajorPosition	28	243	323	594
Premise	152	279	3926	4357
Summe	223	551	4352	5126

Tabelle 17: Konfusionsmatrix für Klassifikator SVM, Datensatz BRK, Subtask B, Features: ①+⑥.

klasse neigt. Dies deckt sich auch mit den Macro F_1-Ergebnissen für CRF auf Subtask B und Datensatz BRK. In diesem Subtask mit drei Klassen schneidet der CRF-Klassifikator spürbar schlechter ab, was an einer Klassifikation der Minderheitsklassen (Positionierungen/Claims) zu den Mehrheitsklassen (Vorschlag/Major Position bzw. Argument/Premise) liegen kann. Diese Annahme bestätigt sich bei genauerer Untersuchung des besten Ergebnisses des Klassifikators CRF für Subtask B (Feature: LDA Wikipedia, Datensatz: BRK) (siehe Tabelle 16). Hier zeigt sich ein deutlich schlechterer F_1-Score für die Minderheitsklassen. Dieses Verhalten beschränkt sich jedoch nicht nur auf den Klassifikator CRF. Auch die Support Vector Machine zeigt für Datensatz BRK und Subtask B schlechtere Ergebnisse (Macro F_1-Score: 51.4%) als auf den restlichen Subtasks. Durch Untersuchung der Konfusionsmatrix (Tabelle 17), wird ebenfalls deutlich, dass die Minderheitsklassen verhältnismäßig stark der Mehrheitsklasse (Premise/Argument) zugeordnet wurden.

Eine mögliche Lösung dieses Problems könnten Oversampling-Verfahren bieten (siehe Kapitel 4.6), weshalb einige exemplarische Experimente mit den in Kapitel 4.6 beschriebenen Oversampling Methoden durchgeführt wurden. Diese umfassen die drei Klassifikatoren RF, SVM und SVM-Linear mit den Oversampling-Verfahren Random Oversampler, SMOTE und ADASYN (mit Standard-Parametern der imblearn-Bibliothek) im Vergleich mit den Ergebnissen ohne Oversampling (siehe Tabelle 18). Leider konnten die Oversampling-Verfahren keine siginifikanten Verbesserungen bewirken. Einzig das Ergebnis des Klassifikators RF konnte durch Hinzunahme des SMOTE-Oversamplers von einem Macro F_1-Score von 45.1% auf 46.1% verbessert werden.

Bezüglich der Datensätze schneidet der Braunkohle-Datensatz generell schlechter ab als der Datensatz zum Tempelhofer Feld. Vor allem Subtask B fällt überdurchschnittlich stark ab. Dies könnte an der deutlich geringeren Anzahl an Vorschlägen im Datensatz

Klassifikator **Oversampler**	RF	SVM	SVM-Linear	Durchschnitt
Default (kein Oversampling)	45.1	51.3	52.7	49.7
Random-Oversampler	45.6	50.9	51.1	49.2
SMOTE	46.1	50.1	51.5	49.2
ADASYN	44.7	51.2	51.6	49.2

Tabelle 18: Oversamplingergebnisse. Es wurde exemplarisch Datensatz BRK, Subtask B und Features ①+②+⑤+⑥ verwendet.

Embedding-Typ	**Trainingskorpus**	**Dim.**	**N-Gramm-Größe**	**Macro F_1-Score**
Character Embeddings	DE-Wikipedia	100	4-4	60.0
	DE-Wikipedia	100	3-6	59.0
	Common Crawl	300	3-6 und 5-5	62.5
Word Embeddings	DE-Wikipedia	100		61.8
	DE-Wikipedia	200		63.6
	DE-Wikipedia	300		63.5

Tabelle 19: Vergleich der Embedding-Typen. Mittelwerte über alle Sentence-Tagging Ergebnisse.

BRK gegenüber THF (siehe Tabelle 10) liegen, was ein Lernen dieser Klasse erschwert. Der kombinierte Datensatz zeigt in den Subtasks A und B vergleichbare Ergebnisse zum Datensatz THF, kann aber in den Subtasks C, D und E deutlich bessere Ergebnisse abliefern. Unter Berücksichtigung von Tabelle 10, wird ersichtlich, dass die beiden Datensätze THF und BRK bezüglich der Verteilung von positiven und negativen Positionierungen sowie Emotionen praktisch invers ausfallen. Der Datensatz BRK weist eine deutlich negativere Grundstimmung auf, was sich in Ablehnungen und negativen Emotionen widerspiegelt, während der Datensatz THF mit Zustimmungen und positiven Emotionen deutlich positiver ausfällt. Durch die Kombination beider Datensätze scheinen die Klassifikatoren eine ausgewogenere Menge an Trainingsinstanzen bezüglich der Ausprägungen von Positionierungen und Emotionen zur Verfügung zu haben, sodass die jeweiligen Ergebnisse besser ausfallen.

Durch Analyse der durchschnittlichen Ergebnisse der einzelnen Features (Tabelle 20), zeigt sich, dass die Kombination ①+②+⑤+⑥ die besten Ergebnisse liefert. Einzig der Klassifikator CRF scheint etwas bessere Ergebnisse bei Verwendung von Character n-grams oder Word Embeddings zu liefern.

Da die Verwendung von Word- und Character-Embeddings auch in späteren Untersuchungen gute Ergebnisse erzielt, wird in Tabelle 19 ein Vergleich der über alle Sentence-Tagging Experimente gemittelten Resultate für die eingesetzten Embedding-Typen geliefert. Hier wird deutlich, dass sich die 300-dimensionalen FastText Embeddings, welche auf dem Common-Crawl-Korpus trainiert wurden, deutlich (4.2%) von den 100-dimensionalen Embeddings des DE-Wikipedia Korpus absetzen können. Ob dieses Ergebnis dem Korpus oder der Dimensionalität geschuldet ist, bleibt für weitere Untersu-

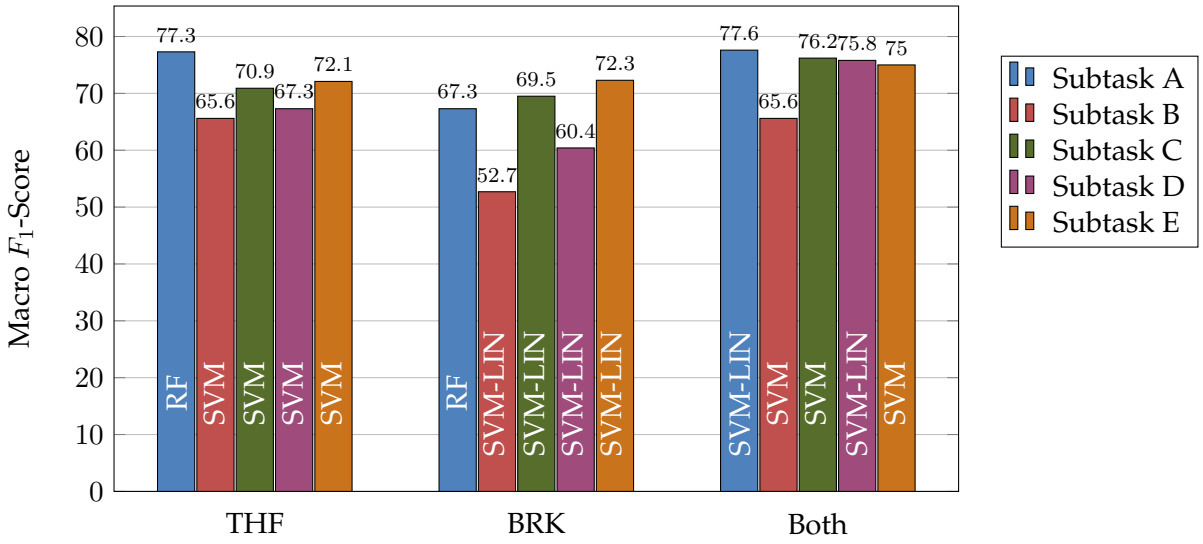


Abbildung 17: Beste Ergebnisse für klassisches Machine Learning.

chungen offen. Die Ergebnisse der Word-Embeddings könnten hier aber einen Hinweis liefern. Diese schneiden mit wachsender Dimensionalität deutlich besser ab und auch liefert die Erhöhung von 100 Dimensionen auf 300 Dimensionen eine Verbesserung um ca. 2.6%. Aufgrund des geringeren Zuwachses könnte daher auf einen Einfluss des Korpus im Falle des Common-Crawl-Korpus geschlossen werden. Weiterhin fällt jedoch auf, dass bereits 200 Dimensionen das beste Ergebnis der Word-Embeddings liefern und die Erhöhung auf 300 keine weitere Steigerung bringt.

Zusammenfassend zeigen sich gute Ergebnisse für alle Subtasks mit wenigen Ausnahmen. Subtask B und D erweisen sich als schwierige Klassifikationsaufgaben auf dem BRK-Korpus, was einem starken Klassenungleichgewicht geschuldet ist. Inhaltlich betrachtet motiviert dies, eine feinere Strukturierung auf Partizipationsplattformen mit dem Vorbild der Beteiligungsplattform zum Tempelhofer Feld vorzunehmen. Hier wurde, im Gegensatz zur Plattform zu Leitentscheidung Braunkohle, das Anlegen von eigenen Vorschlägen ermöglicht. Dies führte zu stringenteren Argumentationen mit klareren Vorschlägen und ausgewogeneren Positionierungen, da häufig Alternativen bzw. eigene Ideen, mit denen sich die Nutzer identifizieren konnten, vorgeschlagen wurden. Die Plattform Braunkohle gab hingegen die Vorschläge in Form von Leitentscheidungen vor, was den organisierten Diskurs und, wie anhand der Ergebnisse zu sehen, auch der Erkennung von Argumentationsstrukturen abträglich war. Als Lösung dieses Problems könnten zukünftige Partizipationsplattformen große Themen wie einzelne Leitschentscheidungen analog zu den Kategorien des THF behandeln, sodass zu jeder Kategorie (hier Leitentscheidung) eigene Vorschläge durch Nutzer erstellt werden können, um eine geordnete Diskussion zu ermöglichen. Dies könnte zu einer Steigerung von positiven emotionalen als auch inhaltlich aussagekräftigen Beiträgen mit respektvollem Umgang und Zustimmungen zu Vorschlägen führen. Eine weiterer Untersuchungen stellt die Erweiterung der Korpora um den jeweils anderen Korpus dar. Durch Kombination beider

Klassifikator	RF		SVM		SVM-Linear		CRF		Durchschnitt	
Durchschnittlicher $F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	64.4	76.5	59.0	76.7	65.2	76.8	54.4	66.3	60.8	74.1
Grammatical ②	60.3	73.3	60.0	71.7	57.8	66.4	57.4	72.4	58.9	71.0
LDA Wikipedia ③	53.3	68.3	54.4	67.4	52.9	65.3	58.7	73.8	54.8	68.7
LDA Dataset ④	56.3	69.7	56.3	68.7	51.5	60.6	58.9	74.0	55.8	68.2
Character n-grams	58.4	74.8	56.6	69.0	56.6	64.3	59.2	74.2	57.7	70.6
Word Embeddings ⑤	61.9	73.8	66.3	76.9	63.7	72.8	59.3	74.2	62.8	74.4
Character Embeddings ⑥	62.1	74.3	65.5	75.8	66.3	74.1	55.6	73.2	62.4	74.3
①+②	61.8	75.0	64.5	78.1	65.9	77.0	57.7	72.6	62.5	75.7
①+③	62.1	75.2	59.8	76.4	65.5	76.4	54.0	66.1	60.4	73.5
①+④	63.0	75.9	65.6	77.9	65.5	76.7	54.3	66.6	62.1	74.3
①+⑤	63.4	75.7	65.0	78.8	66.4	77.5	55.8	69.0	62.7	75.2
①+⑥	62.6	75.4	65.0	75.3	64.2	75.3	54.3	70.7	61.5	74.2
①+②+⑤	63.2	75.7	65.5	78.2	65.9	77.6	57.5	72.2	63.0	75.9
①+②+⑥	62.4	75.8	64.3	75.6	63.3	75.7	56.4	72.6	61.6	74.9
①+②+⑤+⑥	63.5	75.3	65.1	77.6	66.4	77.3	58.2	73.4	**63.3**	**75.9**
Durchschnitt	61.2	74.3	62.2	**74.9**	**62.5**	72.9	56.8	71.4	60.7	73.4

Tabelle 20: Durchschnittliche F_1-Scores des Sentence-Taggings.

Korpora konnten die Ergebnisse einiger Subtasks, besonders Subtask C, D und E, merklich verbessert werden. Dieses Ergebnis motiviert gerade im Fall von Korpora mit einem Bias hinsichtlich der Klassenverteilung in relevanten Subtasks die Nutzung von anderen Datensätzen (idealerweise mit inversem Bias) zur Verbesserung der Klassifikationsergebnisse.

5.1.2 Deep Learning

Im Folgenden werden die Ergebnisse der Deep-Learning Klassifikatoren für die Subtasks auf mittlerer Granularitätsebene (Sentence-Tagging) vorgestellt. Eine Übersicht über die verwendeten Verfahren findet sich in Tabelle 14.

In Tabelle 21 finden sich die Macro- und Micro-F_1-Scores zu allen Subtasks und Datensätzen. Abbildung 18 fasst hierbei die besten Ergebnisse zusammen.

Die Ergebnisse des Deep Learnings fallen durchgängig schlechter aus als jene, welche mit klassischen Machine-Learning-Algorithmen erreicht werden konnten. Dabei fallen die Ergebnisse bezüglich der Subtasks sehr ähnlich aus: Die besten Ergebnisse erzielen Subtasks A, C und E, während Subtasks B und D stark abfallen. Hierbei fällt jedoch ein stärkerer Einbruch der Leistung für Subtask D (Erkennung von Emotionen) auf. Wird hier exemplarisch die Konfusionsmatrix für Datensatz Both und Klassifikator E-LSTM-empty betrachtet (Tabelle 22), so fällt auf, dass viele emotionale Sätze als nicht-emotional klassifiziert wurden und insgesamt ein starkes Klassenungleichgewicht vorherrscht. Im Speziellen ergibt eine Untersuchung der wenigen korrekt als emotional klassifizierten Sätze, dass vor allem Sätze mit vielen Ausrufezeichen erfolgreich trainiert wurden (siehe Tabelle 23). Dies legt nahe, dass im Vergleich zu den klassischen Machine-Learning Algorithmen, welche mit ausgewählten Features getestet wurden, eine stärkere Filterung der Eingabedaten (z.B. Entfernen von Anhäufung von Satzzeichen) nötig ist.

Bezüglich der Klassifikatoren zeigt der Embedding - CNN Klassifikator mit einem durchschnittlichen Macro F_1 von 62.2% die besten Ergebnisse. Auffällig ist, dass der LSTM Klassifikator ohne vorher trainierte Embeddings deutlich besser abschneidet als jener mit extern trainierten Embeddings (vergleiche die Einträge für die Spalten E-LSTM-empty und E-LSTM-pre in Tabelle 21). Dies lässt vermuten, dass ein Anwendungsdomänenspezifisches Training der Embeddings einen deutlichen Vorteil gegenüber allgemeiner trainierten Embeddings liefern kann. Zudem kann es für zukünftige Untersuchungen interessant sein, auch diejenigen Deep Learning Klassifikatoren, welche in dieser Arbeit nur mit vortrainierten Embeddings untersucht wurden, mit einem untrainierten Embedding-Layer zu testen. Auffällig ist zudem, dass sich der E - CNN Klassifikator besonders für diejenigen Subtask mit geringerer Anzahl an Trainingsinstanzen (Aufteilung der Argumente in Subtask B, Unterscheidung der Positionierungen in Subtask C und Unterscheidung der Emotionen in Subtask E) eignet. Die LSTM - basierten Verfahren schneiden hingegen besser für die allgemeine Erkennung von Argumenten (Subtask A) und Emotionen (Subtask D) ab. Vor diesem Hintergrund liegt es nahe in der Analyse von Online-Partizipationsdaten für jeden Subtask einen passenden Klassifikator zu wäh-

Korpus +Subtask		BLSTM		E - CNN		E - CNN - LSTM		E - LSTM - empty		E - LSTM - pre		LSTM - stacked		Durchschnitt	
		Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
BRK	A	55.0	80.8	56.7	**81.2**	**59.4**	80.8	50.1	80.5	58.8	80.6	47.9	80.4	54.6	80.7
	B	41.8	85.6	**43.6**	**85.7**	39.7	85.1	30.6	85.0	37.8	85.2	30.6	85.0	37.4	85.3
	C	48.1	67.7	63.9	70.8	51.4	68.2	**64.8**	**72.4**	39.2	64.6	39.2	64.6	51.1	68.1
	D	48.1	92.8	48.3	92.8	48.1	92.8	**48.8**	**92.8**	48.1	92.8	48.1	92.8	48.2	92.8
	E	63.7	77.9	62.5	**79.8**	49.7	76.9	**64.4**	78.8	44.9	76.0	44.9	76.0	55.0	77.6
Both	A	74.5	**79.4**	74.7	79.3	73.6	78.6	**74.8**	79.2	74.4	79.2	73.8	79.1	74.3	79.1
	B	59.3	75.6	**60.5**	**76.8**	59.0	75.1	48.0	74.5	58.6	75.7	56.1	75.4	56.9	75.5
	C	**74.7**	**76.2**	72.5	75.4	63.8	67.9	72.1	74.2	66.1	67.9	63.9	66.4	68.9	71.3
	D	48.2	93.2	49.1	93.2	50.1	**93.3**	**50.3**	93.2	48.2	93.2	48.2	93.2	49.0	93.2
	E	71.4	72.4	**73.8**	**74.6**	68.6	71.0	72.0	72.5	69.9	71.1	69.5	69.9	70.9	71.9
THF	A	**77.2**	**78.0**	76.9	77.7	77.1	77.9	76.8	77.7	76.3	77.2	76.6	77.2	76.8	77.6
	B	62.1	67.7	**64.6**	**69.0**	63.1	67.1	61.0	65.8	63.2	68.1	61.7	67.9	62.6	67.6
	C	58.3	74.6	**69.5**	**78.0**	51.7	72.6	68.7	75.8	41.9	72.2	41.9	72.2	55.3	74.2
	D	48.3	93.6	48.3	93.6	**61.1**	93.7	50.6	**93.7**	48.3	93.6	48.3	93.6	50.8	93.6
	E	62.9	72.4	**68.1**	**73.4**	57.9	66.7	64.7	69.8	42.6	66.0	40.8	65.8	56.2	69.0
	Durch-schnitt	59.6	79.2	**62.2**	**80.1**	58.3	77.8	59.8	79.1	54.6	77.6	52.8	77.3	57.9	78.5

Tabelle 21: Deep-Learning Sentence-Tagging Ergebnisse. Angegeben werden F_1-Scores

.

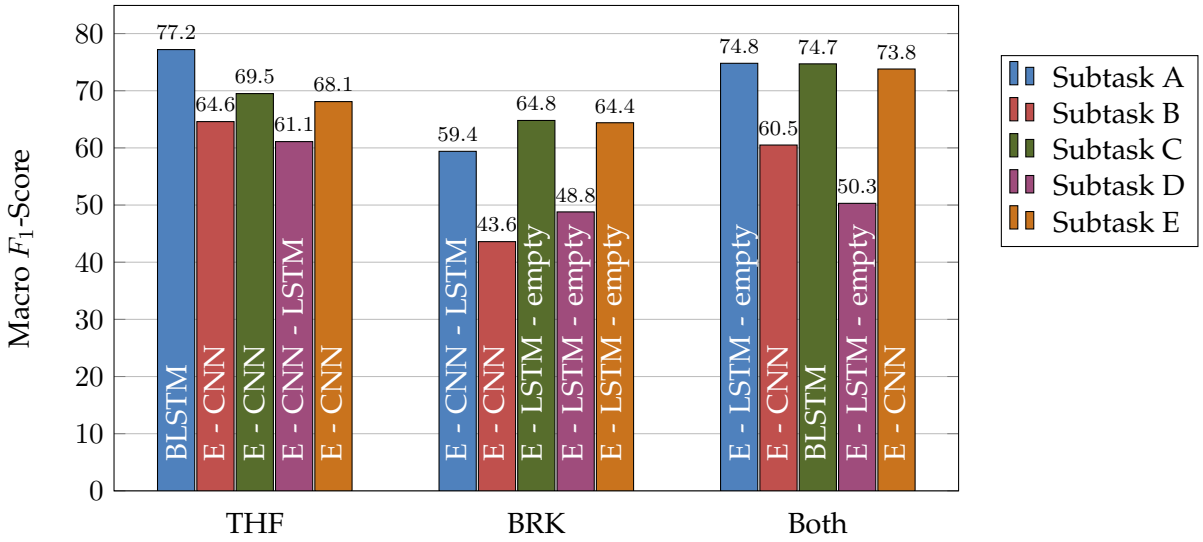


Abbildung 18: Beste Ergebnisse des Sentence-Taggings mit Deep-Learning Verfahren.

len. So könnte eine zukünftige Analyse für die Erkennung von argumentativen Strukturen oder Emotionen einen LSTM - basierten Klassifikator nutzen, die so als argumentativ klassifizierten Datenpunkte dann aber mit einem E - CNN in feinere Klassen aufteilen. Die Gültigkeit der Aussage muss dabei jedoch auf das Anwendungsgebiet des Sentence Taggings eingeschränkt werden. In Kapitel 5.2.3 werden aufgrund des Einflusses der Granularität genauere Folgerungen für die Anwendung dieser Klassifikatoren auf die Anwendung im Sequence Tagging gezogen.

Ein Vergleich der Ergebnisse bezüglich der Datensätze zeigt ein ähnliches Bild zu den klassischen Machine Learning Algorithmen: Datensatz BRK liefert insgesamt schlechtere Ergebnisse, während THF und Both besser abschneiden. Abermals zeigt sich dies insbesondere in Subtask B für Datensatz BRK aus den gleichen Gründen wie bereits in Kapitel 5.1.1 genannt.

Zusammenfassend zeigen die Deep-Learning Verfahren schlechtere Ergebnisse als die klassischen Verfahren. Besonders der Datensatz BRK erweist sich aufgrund des Klassenungleichgewichtes als problematisch. Subtask D (Erkennung von Emotionen) fällt, neben Subtask B, besonders schlecht aus. Hinsichtlich der Klassifikatoren lässt sich die E - CNN Architektur empfehlen, welche aufgrund der ebenfalls guten Ergebnisse der E - LSTM mit leerem Embedding-Layer, in Zukunft ebenfalls ohne vortrainierte Embeddings und gegebenenfalls mit Korpus-spezifisch trainierten Embeddings untersucht werden sollte.

Klassifiziert **Gold**	emotional	non-emotional	Summe
emotional	20	923	943
non-emotional	13	12903	12916
Summe	33	13826	13859

Tabelle 22: Cross-Platform Konfusionsmatrix für den Klassifikator E-LSTM-empty und Datensatz Both.

Klassifiziert: Emotional	**Klassifiziert: Non-emotional**
Hallo hundebesitzer, ihr seid auch gemeint!!!!!!!!!!!!!!!!	verseucht- Hier gab es auch eine sichtbare farbliche Grenzziehung zwischen Grün und braun..
Das lässt sich aber sicherlich auch als gemeinsame Aufgabe lösen !!!	1. Braunkohle dann kommt lange nichts Gleich- Schlimmes
Auch Sie haben ebenfalls wieder vollkommen recht!!!	Daher (egoistisch) dagegen ;-)
Und somit ca 10 Jahre Versorgungssicherheit !!!!!!!!!!!!!!	die kitesurfer machen mir und vielen anderen nutzern angst.
Sofort!!!	das feld ist toll und bunt und frei.
Du willst meine Heimat vernichten!!!!!!!	Wer Gas zur Stromerzeugung verschwendet, begeht ein Verbrechen an unseren Kindern.

Tabelle 23: Beispielsätze für Datensatz Both und Subtask D und Klassifikator E-LSTM-empty. Es werden Sätze mit Gold-Label Emotional aufgelistet.

5.1.3 Klassisches Machine Learning Cross-Platform

In diesem Kapitel werden die Cross-Platform Ergebnisse des Sentence Taggings betrachtet. In allen Tabellen sowie Abbildungen wird jeweils der Datensatz des verwendeten Trainingssets angegeben. Der Datensatz des verwendeten Testsets ist für Cross-Platform Untersuchungen immer der jeweils andere Datensatz. Der kombinierte Datensatz Both wurde hierbei nicht untersucht. Detaillierte Ergebnisse finden sich im Anhang in Kapitel A.1.2. Es folgen die Cross-Platform Ergebnisse für die Klassifikatoren Random Forest (RF), Support Vector Machine (SVM) und Support Vector Machine mit linearem Kernel (SVM-Linear) sowie Conditional Random Fields (CRF).

Die Ergebnisse des Cross-Platform Sentence-Taggings verhalten sich bezüglich der Klassifikatoren und Features ähnlich zu den Sentence-Tagging Ergebnissen aus Kapitel 5.1, fallen jedoch mit wenigen Ausnahmen durchweg schlechter aus. Durchschnittlich kann sich zudem der Klassifikator Random Forest mit einem Makro F_1-Score von 52.1% durchsetzen, wobei die Ergebnisse der beiden SVM-Varianten SVM und SVM-Linear, welche in den normalen Untersuchungen am besten abschneiden konnten, durchschnittlich nur geringfügig schlechter ausfallen und häufig die besten Werte lieferten (siehe Tabelle 23 und Abbildung 24).

Als interessanter erweist sich eine Untersuchung der Ergebnisse pro Subtask. Hierbei fällt auf, dass Subtask A, sobald dieser auf dem Datensatz THF trainiert und Datensatz BRK getestet wird, deutlich schlechter ausfällt (62.2% Macro F_1) als bei umgekehrter Plattformreihenfolge (75.9% Macro F_1). Bei Verwendung von Trainingsset BRK und Testset THF konnten F_1-Scores für die Klassen argumentative und non-argumentative von 81.2% und 70.7% erreicht werden, während bei Verwendung von Trainingsset THF und Testset BRK 86.9% und 37.6% erreicht wurden. Dies lässt vermuten, dass im Fall des Trainings auf dem THF Korpus eine fälschliche Klassifikation von nicht-argumentativen Sätzen als argumentativ vorgenommen wurde. Durch Betrachtung der Konfusionsmatritzen in den Tabellen 25 und 26, bestätigt sich diese Vermutung. Verwunderlich ist jedoch, dass das Testset BRK eine deutlich unausgewogenere Verteilung der Klassen (5122 argumentativ, 1280 nicht argumentativ) besitzt, während das Testset THF mit 4321 argumentativen Sätzen und 3137 nicht-argumentativen Sätzen deutlich ausgeglichener ausfällt (siehe auch Tabelle 10). Da diese Beobachtung umgekehrt dann auch für die Trainingssets gilt, sollten intuitiv genau umgekehrte Ergebnisse zu erwarten sein. Aus diesem Grund ist es verwunderlich, dass die Verfahren, sobald sie auf dem hinsichtlich der Klassenverteilung ausgeglichenen Datensatz trainiert wurden, schlechter abschneiden, als wenn sie weniger Trainingsinstanzen einer Klasse für das Training nutzen konnten. Unter Berücksichtigung von Tabelle 10 könnte davon ausgegangen werden, dass gerade die Erkennung von Sätzen, die zur Klasse Vorschlag (und damit ebenfalls zur Klasse argumentativ) gehören, den Klassifikatoren Schwierigkeiten bereiteten. Mit 633 (38.88%) Vorschlägen im Korpus THF und nur 99 (8.11%) im Korpus BRK könnte dies ein Grund für die unerwarteten Ergebnisse sein, da ein auf dem THF Korpus trainierter Klassifikator stärker auf die Erkennung von Vorschlägen optimiert wurde. Dies würde allerdings nicht die leicht schlechteren Cross-Platform und normalen Ergebnisse für Datensatz BRK in Subtask B erklären. Werden exemplarisch falsche und korrekte Klassifikationen zur Klasse

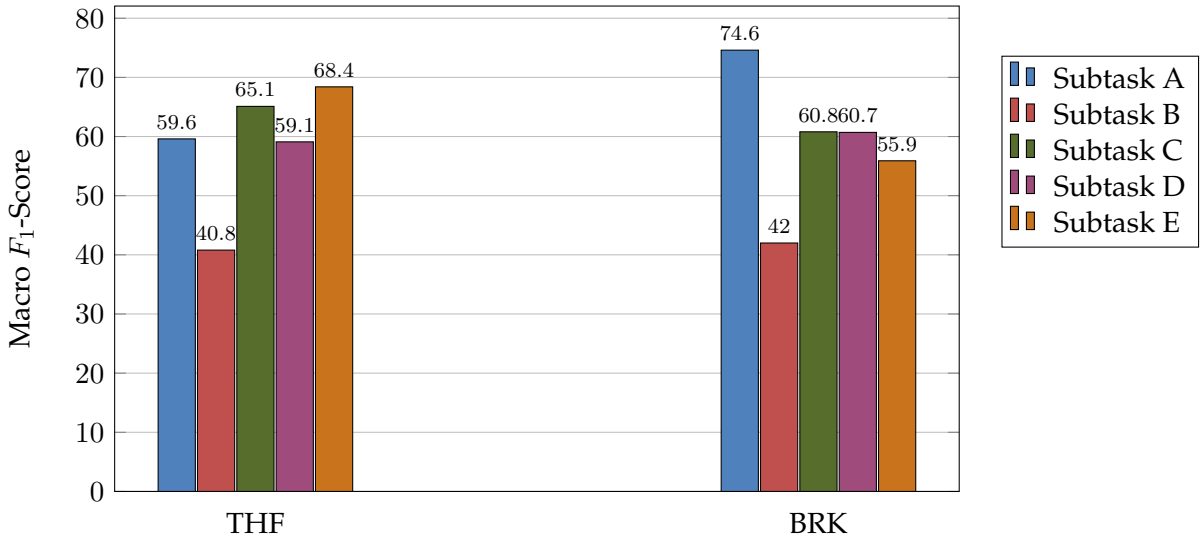


Abbildung 19: Beste Cross-Platform Ergebnisse für den **Klassifikator Random Forest**.

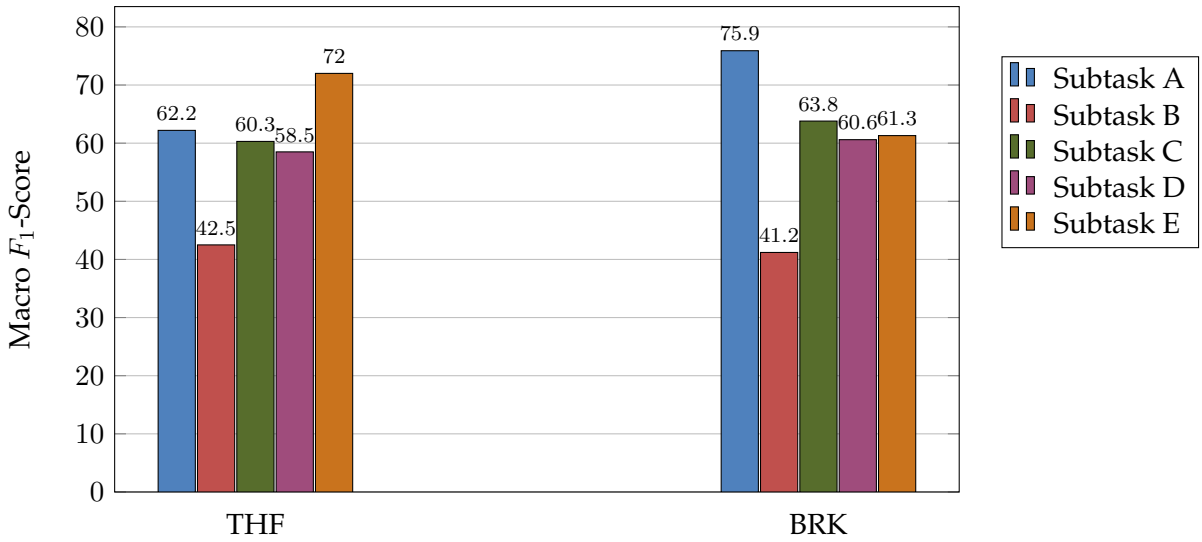


Abbildung 20: Beste Cross-Platform Ergebnisse für den **Klassifikator Support Vector Machine**.

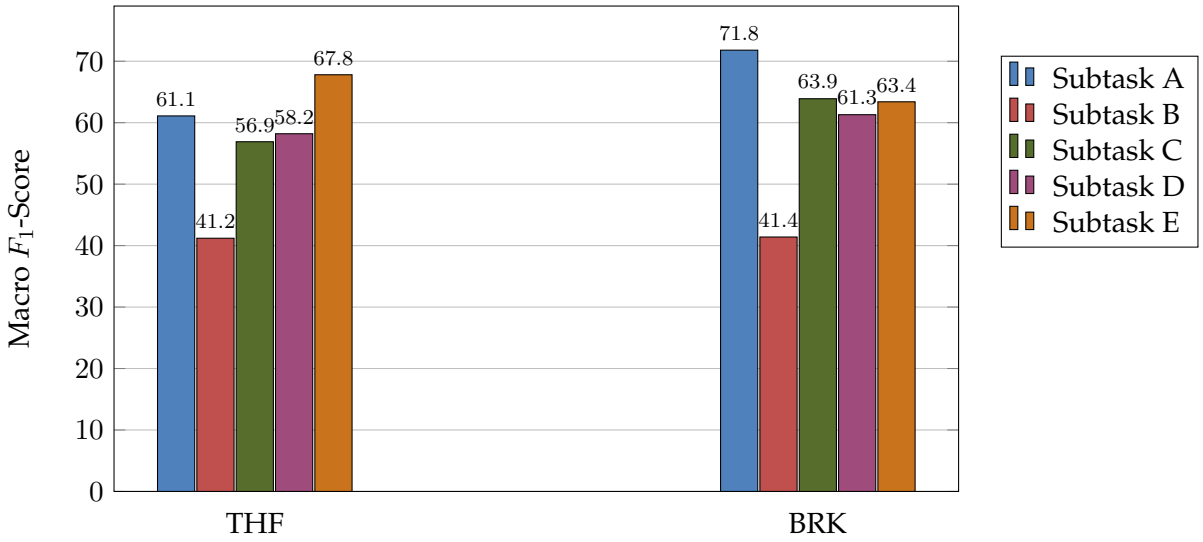


Abbildung 21: Beste Cross-Platform Ergebnisse für den **Klassifikator Support Vector Machine: Linearer Kernel**.

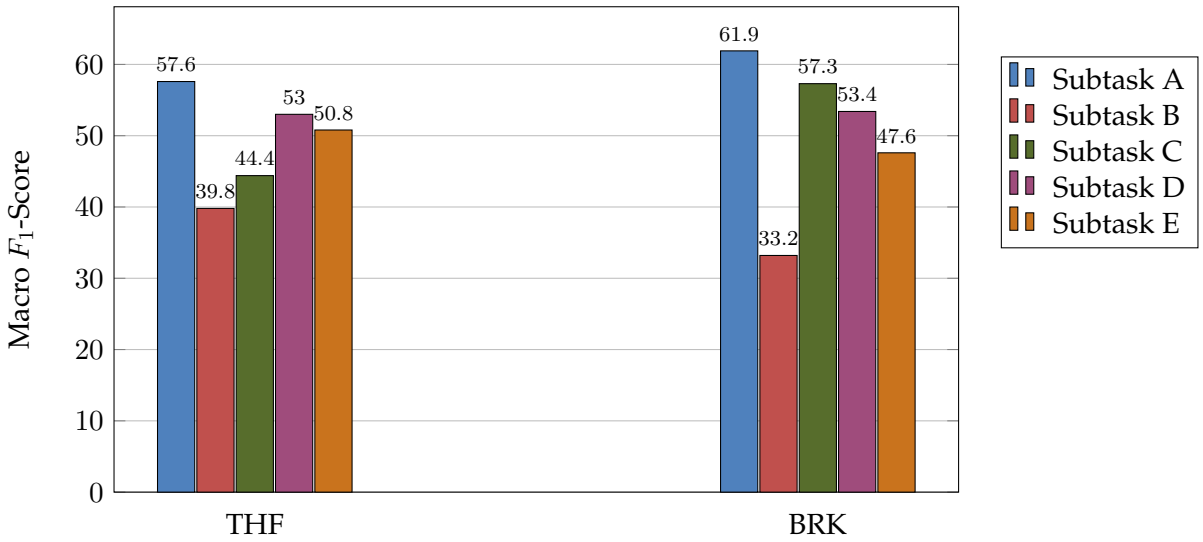


Abbildung 22: BesteCross-Platform Ergebnisse für den **Klassifikator CRF**.

Klassifikator	RF		SVM		SVM-Linear		CRF		Durchschnitt	
Durchschnittlicher $F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	54.1	69.5	46.4	64.7	53.6	66.3	41.2	52.5	48.8	63.2
Grammatical ②	54.3	68.6	51.9	64.5	49.8	58.2	46.8	62.2	50.7	63.4
LDA Wikipedia ③	47.7	60.9	47.5	60.1	42.7	55.8	46.0	60.8	46.0	59.4
LDA Dataset ④	45.3	54.8	47.8	58.2	45.1	51.9	46.3	61.4	46.1	56.6
Character n-grams	48.1	64.2	47.7	58.3	50.4	59.1	46.5	61.6	48.2	60.8
Word Embeddings ⑤	50.7	62.5	54.6	66.6	54.6	65.3	45.5	60.5	51.4	63.7
Character Embeddings ⑥	50.9	63.0	53.6	63.7	51.7	61.7	41.7	58.1	49.5	61.6
①+②	55.1	69.7	49.6	67.2	51.9	63.0	46.5	62.3	50.8	65.5
①+③	52.3	66.0	47.4	63.4	53.8	66.0	42.2	53.3	48.9	62.2
①+④	53.4	67.8	53.7	68.5	54.0	65.8	40.6	52.2	50.4	63.6
①+⑤	54.7	68.7	53.4	69.8	53.7	66.9	43.3	55.2	51.3	65.2
①+⑥	54.5	66.9	55.0	68.5	55.5	69.1	39.5	53.2	51.1	64.4
①+②+⑤	54.7	67.5	52.7	68.3	53.3	67.2	46.8	61.9	51.9	**66.2**
①+②+⑥	52.6	65.2	54.3	68.4	54.7	68.3	45.9	61.8	51.9	65.9
①+②+⑤+⑥	52.4	64.0	54.9	67.9	55.0	68.6	46.7	62.8	**52.2**	65.8
Durchschnitt	**52.1**	**65.3**	51.4	65.2	52.0	63.5	44.4	58.7	49.9	63.2

Tabelle 24: Durchschnittliche Cross-Platform F_1-Scores des Sentence-Taggings.

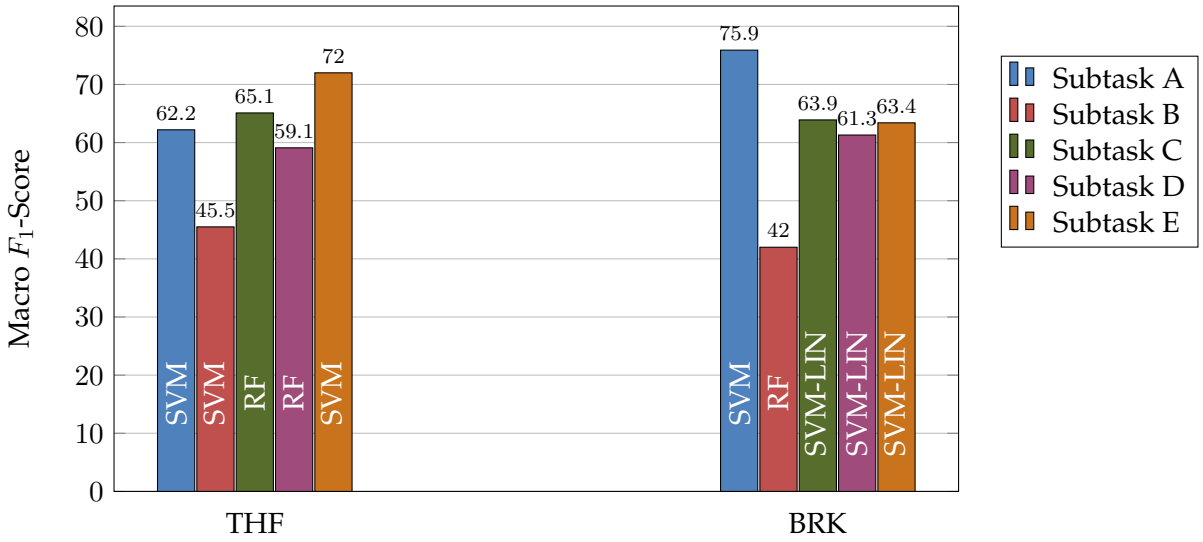


Abbildung 23: Beste Ergebnisse für klassisches Machine Learning mit Cross-Platform Validation.

non-argumentative (Ausschnitt in Tabelle 27) für Subtask A mit Trainingsdatensatz THF und Testdatensatz BRK betrachtet, so fällt auf, dass vor allem längere Sätze, welche logische, aber themenfremde Inhalte wie Metadiskussionen enthalten, fälschlicherweise als argumentativ klassifiziert wurden, während kurze, emotionale oder fragende Sätze meist korrekt als nicht-argumentativ klassifiziert wurden.

Subtask E zeigt ebenfalls große Abweichungen in den Cross-Platform Ergebnissen. Hier entsprechen die Ergebnisse aber aufgrund der Klassenverteilung von positiven und negativen Emotionen (Tabelle 10) weitestgehend den Erwartungen. Auf Datensatz THF trainiert, schneidet der beste Klassifikator (SVM) mit ein Macro F_1 von 72% deutlich besser ab als der beste Klassifikator (SVM-Linear), welcher auf dem Datensatz BRK trainiert wurde und einen Macro F_1-Score von 63.4% liefert.

Insgesamt zeigt sich, dass die Leistung der Klassifikatoren im Cross-Platform Einsatz stark variieren kann, abhängig davon, welcher Datensatz für das Training und den Test genutzt wird. Allgemein fallen die Ergebnisse schlechter aus als bei den Untersuchungen innerhalb eines Datensatzes, dennoch konnte der Datensatz THF eine robuste Trainingsgrundlage für Subtask E und der Datensatz BRK eine robuste Trainingsgrundlage für Subtask A liefern, wie Abbildung 23 zeigt. Datensatz BRK bietet trotz der Probleme in Subtask B eine gute Trainingsgrundlage für Subtask A, ist aber gleichzeitig selbst schwierig zu klassifizieren. Ein Training von Subtask A auf Datensatz BRK und Test auf Datensatz THF schneidet im Vergleich zu den normalen Ergebnissen für Datensatz THF nur geringfügig schlechter ab. Umgekehrt erweist sich der THF-Korpus als verhältnismäßig schlechte Trainingsgrundlage für Subtask A auf Datensatz BRK (vergleiche Abbildungen 17 und 23). Daher liegt es in Zukunft nahe Korpora bezüglich ihrer Schwierigkeit, klassifiziert zu werden, und ihrer Fähigkeit, als Trainingsgrundlage zu dienen, zu unterscheiden. Eine Einschätzung von neuen Korpora bezüglich dieser beiden Eigenschaften könnte anhand eines (oder mehrerer) standardisierten Vergleichskorpus vorgenommen

Klassifiziert **Gold**	argumentative	non-argumentative	Summe
argumentative	4594	528	5122
non-argumentative	862	418	1280
Summe	5456	946	6402

Tabelle 25: Cross-Platform Konfusionsmatrix für Klassifikator SVM, Datensatz THF, Subtask A, Features: ①+⑥.

Klassifiziert **Gold**	argumentative	non-argumentative	Summe
argumentative	3686	635	4321
non-argumentative	1075	2062	3137
Summe	4761	2697	7458

Tabelle 26: Cross-Platform Konfusionsmatrix für Klassifikator SVM, Datensatz BRK, Subtask A, Features: ①+④.

werden. Dieser könnte für zwei Cross-Platform Experimente einmal als Trainingskorpus und einmal als Testkorpus verwendet werden, während der neue Datensatz jeweils als Test- und Trainingskorpus dient. So könnte die Fähigkeit des neuen Korpus als Trainingsgrundlage zu dienen und die Schwierigkeit diesen zu klassifizieren anhand der Ergebnisse der Experimente mit dem Standardkorpus mit anderen Korpora verglichen werden. Dieses Verfahren setzt dabei jedoch das Bestehen von Gold-Labels für den zu untersuchenden Korpus voraus.

Klassifiziert: Non-argumentative	**Klassifiziert: Argumentative**
Danke!	Ich bitte das Gremium die Entscheidungen treffen und wahrscheinlich nicht um Ihren Arbeitsplatz bangen müssen oder Hohe Stromkosten bezahlen müssen , meine geschriebenen Worte mitzunehmen.
Warmes Wasser von April bis Oktober auch von der Sonne Als nächstes ist die Heizung dran....	Solange man für hierfür keine hieb- und stichfesten Erklärungen anführen kann, sollte man sich mit deratigen Äußerungen ein wenig zurückhalten.
Welche Alternativen haben wir?	Das Team der Moderation hatte darum gebeten möglichst konkrete Vorschläge für die Formulierung des Leitsatzes zu machen, dem möchte ich hier nachkommen:
Auch Sie haben ebenfalls wieder vollkommen recht!!!	Liebe Teilnehmerinnen und Teilnehmer, vielen Dank für den Lesestoff, Hintergrundinformationen und Ihre Einschätzungen zur Energie- und Klimapolitik.
Auf wessen Gehaltsliste stehen Sie eigentlich?	Heute Nachmittag lief eine ausgesprochen interessante, aber auch furchteinflößende Dokumentation im TV.

Tabelle 27: Cross-Platform Beispielsätze für Trainingsdatensatz THF und Testdatensatz BRK, Subtask A. Es werden Sätze mit Gold-Label non-argumentative aufgelistet.

5.1.4 Deep Learning: Cross-Platform

Die Ergebnisse des Cross-Platform Sentence Taggings mit Deep Learning Klassifikatoren fällt analog zu den Experimenten innerhalb der Datensätze entsprechend schlechter aus als die Ergebnisse der klassischen Machine Learning Verfahren. Gleichzeitig wiederholen sich die Besonderheiten der Ergebnisse: Datensatz BRK liefert eine deutlich bessere Trainingsgrundlage für Subtask A während beide Datensätze schlechtere Ergebnisse auf einem fremden Testset liefern als auf dem gleichen Datensatz. Hinsichtlich der Klassifikatoren kann sich der Embedding LSTM Klassifikator ohne vortrainierte Embeddings durchsetzen (siehe Tabelle 31). Im Vergleich zum Embedding - CNN Netz kann sich das E - LSTM - empty Netz vor allem in Subtask C auf Datensatz BRK durchsetzen. Werden hierzu die Konfusionsmatritzen der beiden Klassifikatoren (Tabellen 28 und 29) verglichen, so zeigt sich, dass der E - CNN Klassifikator in diesem Fall viele Sätze fälschlicherweise als positive Positionierungen klassifiziert, während der E - LSTM - empty Klassifikator zwar eine schlechtere Precision bezüglich Positionierungen Pro aufweist, allerdings dafür deutlich besser negative Positionierungen erkennt. Überraschenderweise wiederholt sich das gute Ergebnis für Subtask E auf Trainingsdatensatz THF, welches in Abbildung 23 beobachtet werden konnte, nicht. Das Ergebnis fällt hier nur leicht besser als auf Trainingsdatensatz BRK aus. Im Vergleich des Ergebnis des E - LSTM - pre (Macro F_1-Scores von 37.7 für EMOP und 80.7 für EMON) mit dem besten Ergebnis der klassischen Machine-Learning Verfahren (SVM mit Word Embeddings, Macro F_1: 87.4 für EMOP und 56.6 für EMON), fällt auf, dass das Embedding LSTM Netz vor allem Schwierigkeiten hatte, positive Emotionen zu erkennen, und klassifizierte diese häufig als negative Emotion. Werden dazu Beispielsätze betrachtet, welche von der SVM korrekt klassifiziert wurden, vom E - LSTM - pre Klassfikator aber falsch klassifiziert wurden (Tabelle 30), so fällt auf, dass vor allem längere, komplexe Sätze mit indirekt geäußerten Emotionen ein Problem darstellen. Hierbei ist es umso überraschender, dass die verwendete SVM in der Lage war, diese Sätze korrekt zu klassifizieren. Durch Untersuchung von Tabelle 61 wird jedoch deutlich, dass dieses Ergebnis nur durch die Verwendung von Word Embeddings (im Speziellen 300 dimensionale DE-Wiki Word2Vec Embeddings) erzielt werden konnte, was erneut die Untersuchung von verschiedenen Konfigurationen des Embedding Layers der Deep Learning Verfahren motiviert.

Klassifiziert **Gold**	Position pro	Position contra	Summe
Position pro	276	22	298
Position contra	104	11	115
Summe	380	33	413

Tabelle 28: Konfusionsmatrix für Sequence Tagging Subtask C, Klassifikator E - CNN

Klassifiziert Gold	Position pro	Position contra	Summe
Position pro	169	129	298
Position contra	27	88	115
Summe	196	217	413

Tabelle 29: Konfusionsmatrix für Sequence Tagging Subtask C, Klassifikator E - LSTM - empty

Klassifiziert: Positive Emotion	**Klassifiziert: Negative Emotion**
Lag wohl auch an unserer Braunkohle ;-)	Ich vertraue fest darauf, jeder der Teilnehmer genau weiß, was uns bevorsteht und rechne mit dem Setzen neuer Ziele für unser Klima, die einen umgehenden Stopp der Braunkohleförderung zwingend notwendig machen.
Gott sei Dank!	Dies können Sie natürlich auch an den vielen schönen Seen im Brühler Revier tun.
Es ist die Zeit gekommen!	Wandern Sie für einige erholsame Stunden auf der Sophienhöhe bei Hambach.
mit diesem Beitrag sprechen Sie mir zu 100% aus der Seele	Deshalb bin ich zuversichtlich, dass sich auch der Garzweiler See zu einem Gewinn für die Landschaft entwickeln wird - wie die vielen bereits vorhandenen Seen im rheinischen Braunkohlenrevier (Ville-Seen, Blausteinsee, Otto-Maigler-See etc.) und in den ostdeutschen Revieren (Cospudener See, Senftenberger See, Goitzsche-See, Geiseltalsee etc.).

Tabelle 30: Cross-Platform Beispielsätze für Trainingsdatensatz THF, Subtask E und Klassifikator E-LSTM-pre. Es werden Sätze mit Gold-Label Positive Emotion aufgelistet. Die falsch klassifizierten Sätze (rechte Spalte) konnten von einer SVM korrekt klassifiziert werden.

Korpus +Subtask		BLSTM		E - CNN		E - CNN - LSTM		E - LSTM - empty		E - LSTM - pre		LSTM - stacked		Durchschnitt	
		Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
BRK	A	74.5	76.3	74.1	74.6	38.9	58.1	75.4	76.8	**75.9**	**76.9**	74.7	76.3	68.9	73.2
	B	35.2	54.2	36.8	**55.3**	27.7	53.9	**38.8**	53.7	27.8	53.6	28.8	53.7	32.5	54.1
	C	52.3	52.5	48.1	**69.5**	55.2	56.2	**60.7**	62.2	21.8	27.8	21.8	27.8	43.3	49.3
	D	48.3	93.6	48.3	93.6	**50.0**	93.6	48.8	93.6	48.3	93.6	48.8	**93.6**	48.8	93.6
	E	57.3	57.4	**58.7**	**61.8**	36.6	40.5	58.6	59.3	27.0	34.8	27.4	35.0	44.3	48.1
THF	A	53.8	80.5	56.1	80.4	53.8	80.3	54.8	80.1	55.2	**80.7**	**56.5**	80.3	55.0	80.4
	B	31.8	83.7	31.4	84.3	31.1	83.3	**34.5**	82.6	30.6	85.0	30.6	**85.0**	31.7	84.0
	C	52.4	52.6	**53.9**	**54.7**	27.9	36.5	49.5	50.5	28.5	36.5	26.2	35.4	39.7	44.4
	D	48.1	92.8	48.6	92.8	48.3	92.8	**51.4**	92.5	48.1	92.8	48.1	**92.8**	48.8	92.8
	E	55.6	**76.9**	50.0	76.2	56.9	66.9	53.6	64.1	**59.2**	70.5	55.4	67.9	55.1	70.4
Durch-schnitt		50.9	72.0	50.6	**74.3**	42.6	66.2	**52.6**	71.5	42.2	65.2	41.8	64.8	46.8	69.0

Tabelle 31: Cross-Platform Deep-Learning Sentence-Tagging Ergebnisse.

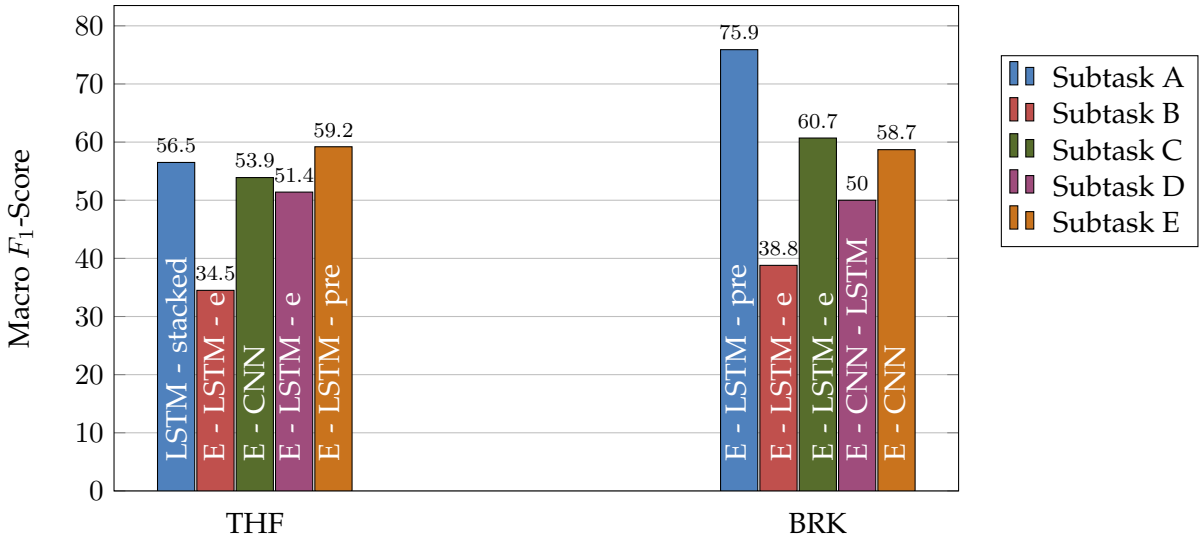


Abbildung 24: Beste Cross-Platform Ergebnisse des Sentence-Taggings mit Deep-Learning Verfahren.

5.2 Sequence-Tagging

5.2.1 Parameter

Für die Durchführung des Sequence-Taggings ist es nötig, vor der Evaluation zusätzlich zu den im Sentence-Tagging verwendeten Hyperparametern die, in Kapitel 4.5.3 eingeführten, Hyperparameter festzulegen. Diese wurden wie folgt gewählt:

Für das Padding wurde Zero-Padding verwendet. Zur Bestimmung der Fenstergröße, d.h. der Anzahl der jeweils vor und nach dem aktuellen Token verwendeten Token zur Konstruktion von „künstlichen Sätzen“ (siehe Kapitel 4.5.3), und der Gewichtung des mittleren Tokens wurde ein Gridsearch durchgeführt (siehe Tabelle 32). Es zeigt sich, dass die Gewichtung des mittleren Tokens nur eine untergeordnete Rolle bezüglich des F_1-Scores spielt, während die Fenstergröße einen signifikanten Einfluss besitzt. So verbessert sich das Ergebnis für Gewichtungswert 1 von Fenstergröße 1 zu 8 um 2,2 Prozentpunkte (6% Verbesserung des Macro F_1-Scores). Die Verbesserung der Ergebnisse nimmt für steigende Fenstergrößen dabei immer weiter ab, was sich damit begründen lässt, dass die Satzlängen in Online-Diskussionen meist kürzer ausfallen als in formalen Texten, sodass höhere Fenstergrößen dazu neigen, häufig Sätze mit hohem Padding-Anteil zu generieren. Aufgrund dieser Ergebnisse wird fortan für alle Sequence-Tagging Aufgaben die Fenstergröße 8 und Gewichtung 2 verwendet, in Zukunft jedoch aus Effizienzgründen Fenstergröße 6 und Gewichtung 2 empfohlen.

Zusätzlich muss eine Wahl für das verwendete Sequenzannotationsschema getroffen werden. Im Fall von Token-basierter Klassifikation ist es wichtig ein Annotationschema zu verwenden, welches eine Trennung zweier aneinanderliegender Sequenzen erlaubt, falls diese der gleichen Klasse angehören. Würden nur die normalen Klassen als Label verwendet werden, wäre diese Trennung nicht möglich. Aus diesem Grund wird in Sequence-Tagging Applikationen wie z.B. der Named Entity Recognition häufig das BIO-Schema (Ramshaw und Marcus, 1999) verwendet. Während der Erstellung dieser Arbeit fiel auf, dass spaCy[20] in den eigenen Sequence-Tagging Verfahren das BILOU-Schema verwendet, welches das BIO-Schema durch die Bezeichner L (Last - letztes Element einer Sequenz) und U (Unit - alleinstehendes Token) erweitert. Nach Ratinov und Roth (2009)

[20] https://spacy.io/api/annotation

Gewichtung	**Fenstergröße**					
	1	2	3	4	6	8
1	36.9	37.8	38.1	38.7	39.1	39.1
2	36.9	37.9	38.2	38.5	39.1	**39.2**
3	36.9	37.9	38.2	38.5	39.0	**39.2**

Tabelle 32: Macro F_1 Gridsearch-Ergebnisse für Sequence-Tagging: Gewichtung und Fenstergröße für die Erstellung künstlicher Sätze. Es wurde exemplarisch Subtask A für den gemischten Datensatz mit den Features Unigram+Grammatical+Word Embeddings+Character Embeddings mit Klassifikator SVM untersucht und nur die normalen Klassenlabels verwendet (kein BIO oder BILOU-Schema).

Schema	Klassifikatoren							
	RF		SVM		SVM-Linear		Durchschnitt	
	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
BIO	28.4	55.4	30.4	57.3	26.5	41.4	28, 4	51.4
BILOU	15.4	52.4	20.6	52.6	15.9	37.5	17.3	47.5
Default	55.4	57.0	56.7	58.2	54.8	55.3	55.6	56.8
BIO (Default)	56.4	58.2	58.4	60.3	55.7	56.6	57.1	58.4
BILOU(Default)	55.5	57.6	57.6	59.4	55.3	56.5	56.1	57.2

Tabelle 33: Macro F_1 Gridsearch-Ergebnisse für Sequence-Tagging: Einfluss der Annotationschemata auf die Klassifikationsgenauigkeit. Es wurde exemplarisch Subtask A für den gemischten Datensatz mit dem Feature Unigram untersucht. Schema Default verwendet nur die originalen Klassenlabels.

kann dieses Schema die Klassifikationsergebnisse verbessern. Aus diesem Grund wurde die Klassifikationsleistung für die beiden Schemata exemplarisch auf dem Datensatz Both mit Subtask A und für die Klassifikatoren RF, SVM und SVM-linear verglichen (siehe Tabelle 33. Zusätzlich zu den Schemata Default, BIO und BILOU wurden die beiden Schemata BIO und BILOU mit anschließender Zusammenfassung der Klassenlabels auf die Klassenlabels aus dem Default-Schema (nur originale Klassenlabels) untersucht. Dazu wurden die durch die Schemata erzeugten Prefixe (B,I bzw. B,I,L,U) entfernt. Klasse O war aufgrund des verwendeten Subtasks (A) nicht vorhanden, müsste jedoch je nach Einschränkung des Datensatzes für den jeweiligen Subtask adäquat ersetzt werden. Hierbei wird deutlich, dass das BIO-Schema in diesen Ergebnissen deutlich bessere Ergebnisse im Sequence-Tagging liefert als das BILOU-Schema, weshalb in den folgenden Sequence Tagging Experimenten immer das BIO-Schema verwendet wurde. Außerdem liefert das BIO-Schema auch zur Erzeugung von Default-Labeln durchgängig bessere Ergebnisse als die Verwendung des Default-Schemas, was aufgrund der höheren Anzahl an zu trainierenden Klassen überraschend ist.

Aufgrund der durchweg schlechteren Ergebnisse sowie der Kombination aus hohem Speicherbedarf und langer Laufzeit wurde der Klassifikator CRF in den Sequence Tagging Experimenten nicht weiter untersucht. Damit konnten die in Goudas et al. (2014) vorgenommen Experimente mit Conditional Random Fields auf feiner Granularitätsebene nicht wiederholt werden, was jedoch angesichts der Ergebnisse der hier untersuchten Variante auf den anderen Granularitätsebenen nur eine geringe Wahrscheinlichkeit für bessere Resultate liefern würde. Es wird jedoch in Zukunft empfohlen die CRF Architektur unter Verwendung einer effizienteren Implementierung erneut auf feingranularen Daten zu testen.

5.2.2 Klassisches Machine Learning

Es ist anzumerken, dass im Fall des Sequence Taggings keine vorherige Filterung der Daten bezüglich der Subtasks durchgeführt wurde, sodass ein großer Teil der Daten dem „O“-Label zugeordnet wurde. So wurden z.B. die Trainings und Testinstanzen für Sub-

task B nicht auf argumentative Textteile reduziert. Grund dafür war die Annahme, dass die Klasse „O“ des BIO-Schemas adäquate F_1-Scores aufweisen würde, was eine Vergleichbarkeit zu anderen Sequence-Tagging Verfahren liefern könnte. Dies schränkte leider die Aussagekraft der Ergebnisse für Subtasks B, C und E im Vergleich zu den Experimenten auf mittlerer und grober Granularitätsebene ein. Eine Wiederholung der Sequence Tagging Experimente mit Einschränkung der Korpora für die jeweiligen Subtasks wäre daher anzuraten. Dennoch liefern die Experimente für Subtask A und D aussagekräftige und vergleichbare Resultate, da hier binäre Klassifikationen auf den ungefilterten Korpora durchgeführt werden.

Die Ergebnisse des Sequence Taggings mit klassischen Machine Learning Verfahren fallen deutlich schlechter aus als die Resultate auf mittlerer und grober Granularitätsebene. Auffällig hierbei sind die in Relation zu den anderen Subtasks guten Resultate für Subtask D (Erkennung von Emotionen), welche mit Macro-F_1 Scores von 29.3%, 29.8% und 29.4% für die Datensätze THF, BRK und Both ähnlich gut ausfallen wie die Ergebnisse für Subtask A (siehe Abbildung 28). Subtask B liefert, wie anhand der vorherigen Ergebnisse aufgrund des Klassenungleichgewichts zu erwarten war, die schlechtesten Ergebnisse (insbesondere für Datensatz BRK), während Subtasks C und E bessere Ergebnisse auf Datensatz THF liefern als auf BRK und Both. Werden dazu die Konfusionsmatritzen der besten Resultate für Subtask C auf den Datensätzen BRK und THF (Tabellen 34 und 35) betrachtet, so spiegelt sich die in Tabelle 10 sichtbare Verteilung von positiven und negativen Positionierungen direkt in den Klassifikationsergebnissen wider: Die Klassifikatoren neigen dazu auf Datensatz BRK positive Positionierungen als negative Positionierung zu klassifizieren, während die Klassifikation von Positionierungen auf Datensatz THF deutlich ausgeglichener ist. Gleichzeitig fällt die Erkennung von Argumenten und Emotionen (Subtasks A und E) auf Datensatz BRK verhältnismäßig gut aus. Dies deckt sich mit den Ergebnissen zu Subtask B auf Datensatz BRK in den Document Tagging Untersuchungen (siehe Kapitel 5.3.1, Abbildung 39). Eine genauere Analyse zu diesem Verhalten findet sich in Kapitel 5.3.1.

Im Vergleich der Ergebnisse bezüglich der Klassifikatoren, liefert die SVM die besten Ergebnisse, kann aber gerade in dem sehr gut abschneidenden Subtask A auf Datensatz BRK durch den Klassifikator RF geschlagen werden. Dieses Verhalten kann ebenfalls für die Untersuchungen des Document Taggings in Kapitel 5.3.1 beobachtet werden und wird dort weiter kommentiert.

Bezüglich der verwendeten Features liefert die Kombination ①+⑥ (Unigram + Character Embeddings) die besten Ergebnisse, wobei die in den meisten Untersuchungen am besten abschneidende Kombination ①+②+⑤+⑥ nur um 0.1 davon abweicht. Insgesamt kann also erneut eine Empfehlung für die Verwendung von Character Embeddings, insbesondere den FastText Embeddings, welche auf dem CommonCrawl Korpus trainiert wurden, sowie Word Embeddings (200 oder 300 dimensionale Word2Vec Embeddings des DE-Wikipedia Korpus) ausgesprochen werden. Besonders schlecht schnitten die LDA-Features ab. Unter Berücksichtigung der Untersuchungen auf mittlerer und grober Granularitätsebene (Kapitel 5.1 und 5.3) ergeben sich unterdurchschnittliche bis schlechte Ergebnisse für die Verwendung der LDA-Features. Aus diesem Grund wird fortan von einer alleinigen Verwendung dieser abgeraten. Eine Untersuchung der Kombination von LDA-Features und weiteren Features (hier: Unigram-Verteilung) kann je-

Klassifiziert Gold	B_ClaimContra	B_ClaimPro	I_ClaimContra	I_ClaimPro	O	Summe
B_ClaimContra	2	0	2	0	82	86
B_ClaimPro	1	4	7	1	98	111
I_ClaimContra	3	7	**71**	**58**	1099	1238
I_ClaimPro	6	7	**252**	**118**	2567	2950
O	121	88	1780	952	56112	59053
Summe	133	106	2112	1129	59958	63438

Tabelle 34: Konfusionsmatrix für Sequence Tagging Klassifikator SVM, **Datensatz BRK**, Subtask C, Features: ①+②+⑤+⑥.

Klassifiziert Gold	B_ClaimContra	B_ClaimPro	I_ClaimContra	I_ClaimPro	O	Summe
B_ClaimContra	4	1	3	1	115	124
B_ClaimPro	0	22	0	22	209	253
I_ClaimContra	0	3	**60**	**28**	1195	1286
I_ClaimPro	0	44	**5**	**179**	1969	2197
O	16	110	83	225	38127	38561
Summe	20	180	151	455	41615	42421

Tabelle 35: Konfusionsmatrix für Sequence Tagging Klassifikator SVM, **Datensatz THF**, Subtask C, Features: ①+⑤.

doch empfehlenswert sein, wie anhand der Ergebnisse in Tabelle 36 deutlich wird. Hier konnte eine Kombination von Unigram und auf dem Datensatz trainierter LDA einen Macro F_1-Score von 24.2% (SVM) erreichen, während Unigram bzw. LDA Dataset alleine respektive 21.7% und 8.6% lieferten.

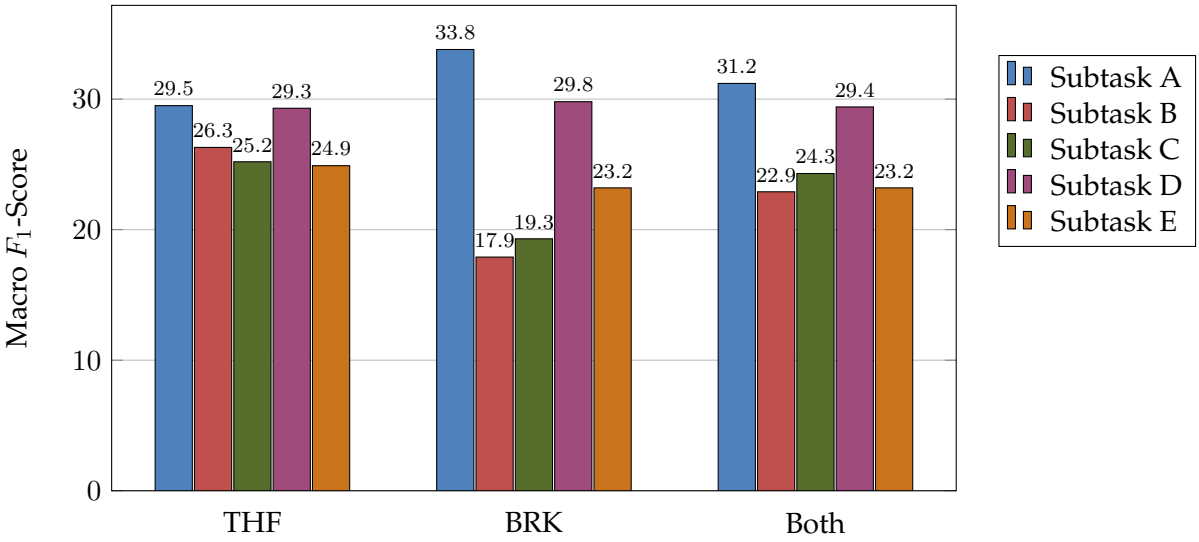


Abbildung 25: Beste Ergebnisse für den **Klassifikator Random Forest**.

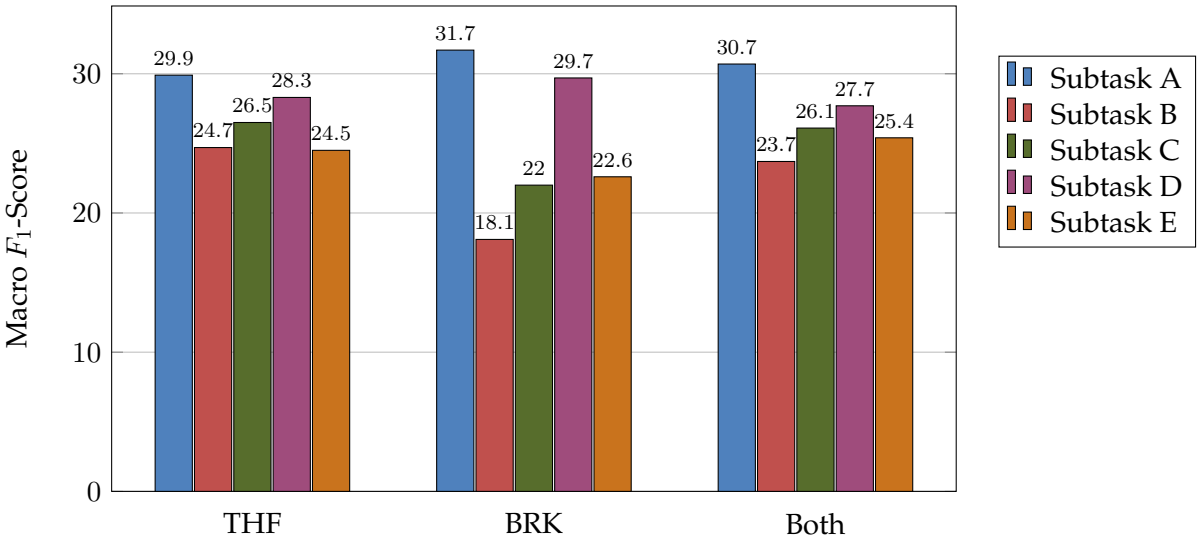


Abbildung 26: Beste Ergebnisse für den **Klassifikator Support Vector Machine**.

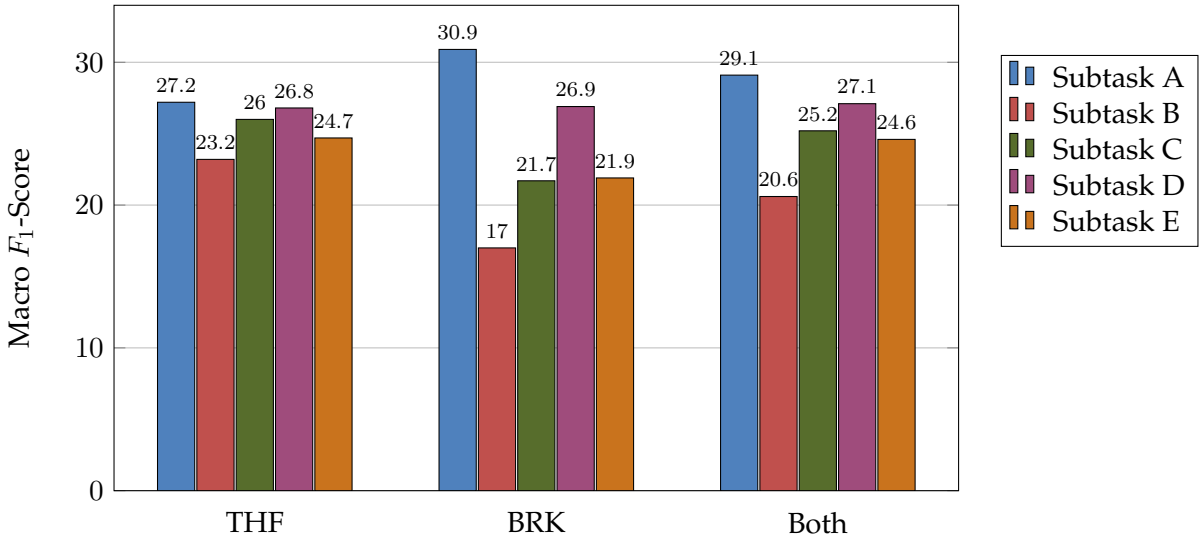


Abbildung 27: Beste Ergebnisse für den **Klassifikator Support Vector Machine: Linearer Kernel**.

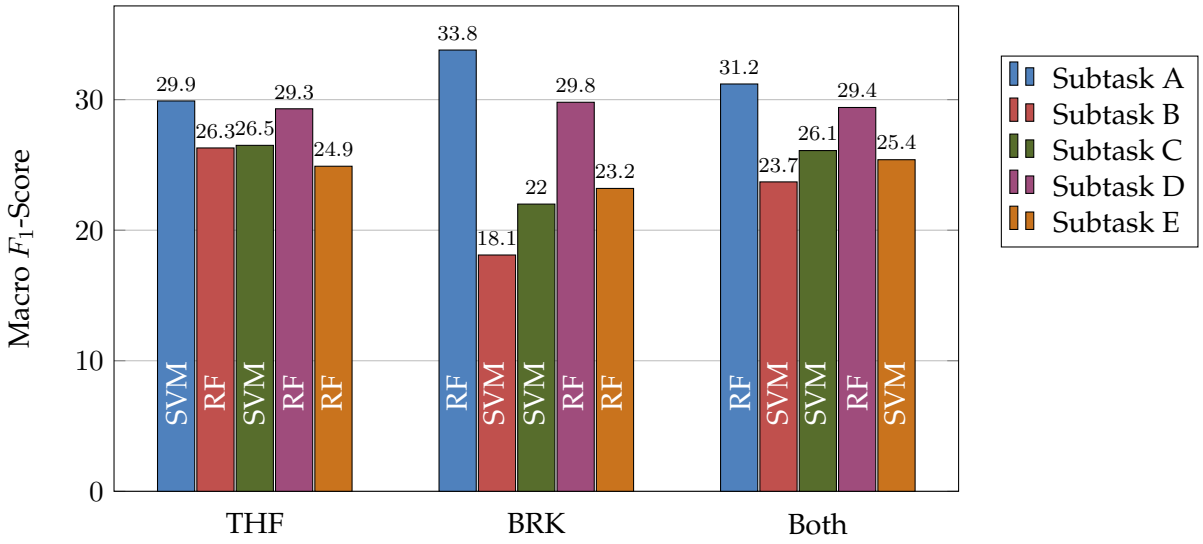


Abbildung 28: Beste Ergebnisse des Sequence Taggings mit klassischen Machine Learning Verfahren.

Klassifikator	RF		SVM		SVM-Linear		Durchschnitt	
Durchschnittlicher $F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	21.4	46.7	21.7	64.0	23.0	54.1	22.0	54.9
Grammatical ②	22.7	53.5	20.0	39.4	16.1	25.6	19.6	39.5
LDA Wikipedia ③	9.9	25.9	7.9	19.3	12.6	37.7	10.1	27.6
LDA Dataset ④	8.0	12.7	8.6	16.3	11.7	32.2	9.4	20.4
Character n-grams	23.8	63.8	21.9	53.0	19.4	32.1	21.7	49.6
Word Embeddings ⑤	20.5	51.0	22.5	51.9	18.3	32.9	20.4	45.3
Character Embeddings ⑥	20.8	42.5	23.6	50.0	21.2	36.4	21.9	43.0
①+②	21.2	45.9	24.0	62.2	23.2	53.7	22.8	53.9
①+③	20.0	41.3	22.6	61.4	22.5	52.5	21.7	51.7
①+④	22.5	49.0	24.2	61.6	22.6	54.0	23.1	54.9
①+⑤	20.8	49.3	22.3	64.8	22.8	54.3	22.0	**56.1**
①+⑥	22.3	51.6	24.9	56.7	24.0	54.9	**23.7**	54.4
①+②+⑤	20.9	47.8	24.5	63.3	23.3	55.2	22.9	55.4
①+②+⑥	20.9	44.9	24.9	56.8	24.4	55.7	23.4	52.5
①+②+⑤+⑥	20.5	42.2	25.6	61.6	24.6	56.5	23.6	53.4
Durchschnitt	19.7	44.5	**21.3**	**52.2**	20.6	45.9	20.6	47.5

Tabelle 36: Durchschnittliche F_1-Scores des Sentence-Taggings.

5.2.3 Deep Learning

Die Sequence Tagging Ergebnisse mit Deep Learning Verfahren liefern erstmals in den Experimenten bessere Resultate als die klassischen Machine Learning Verfahren. Hierbei sind vor allem die überraschend guten Ergebnisse für Subtasks A und C (Unterscheidung von Positionierungen) zu nennen. Aufgrund der fehlenden Filterung der Daten war insbesondere für Subtask C ein schlechteres Ergebnis zu erwarten, da die Subtasks B, C und E in den vorherigen Untersuchungen ein starkes Ungleichgewicht zur Klasse „O“ aufwiesen (siehe auch Konfusionmatrix 34). Dennoch kann insbesondere der BLSTM-Klassifikator auf Subtask C sehr gute Ergebnisse erzielen. Dies motiviert die Verwendung dieser Ergebnisse für Intergranularitätsuntersuchungen (siehe Kapitel 5.4).

Bezüglich der Klassifikatoren liefert der BLSTM-Klassifikator durchschnittlich die besten Resultate. In einigen Subtasks (z.B. Subtask A auf Datensatz THF) kann jedoch der E-CNN Klassifikator bessere Ergebnisse liefern. Insgesamt liefern vor allem die in den anderen Experimenten dominanten Embedding-LSTM Klassifikatoren hier schlechtere Ergebnisse. Es scheint, dass eine größere Anzahl an Trainingsdaten kombiniert mit einer schwierigeren Klassifikationsaufgabe besonders gut für die BLSTM-Architektur geeignet ist. Dieser Eindruck wird durch die Ergebnisse in Abbildung 18 der Sentence Tagging Untersuchungen verstärkt. Hier kann das BLSTM-Netz ebenfalls auf Datensatz THF (viele, kurze Kommentare) besser abschneiden als die anderen Deep-Learning Klassifikatoren. Werden zusätzlich die Cross-Platform Ergebnisse des Sequence Taggings (Abbildung 34) betrachtet, kann davon ausgegangen werden, dass die Stärke des BLSTM-Netzes vor allem im Training vieler feingranularer, schwieriger Daten liegt, da die Leistung bei Verwendung des Trainingsdatensatzes THF und Testdatensatz BRK weiterhin hoch bleibt. Gegenüber der ebenfalls gut abschneidenden E-CNN Architektur kann der BLSTM-Klassifikator besonders in Subtask C (Unterscheidung von Positionierungen) bessere Resultate liefern. Hier liegen weniger Datenpunkte für das Training vor, die Klassifikation bleibt aber aufgrund des Sequence Taggings schwierig. Im Gegensatz dazu kann im Fall des Sentence Taggings in Tabelle 21 ein Vorteil für den E-CNN Klassifikator in Subtask C beobachtet werden. Dieser wird mit wachsender Korpusgröße kleiner. Dieses Verhalten spiegelt zum Teil den Unterschied der Architekturen wieder: Während Convolutional Neural Nets lokale Informationen sehr gut verarbeiten können (geeignet für gröbere Granularität und einheitlichen Datensatz), berücksichtigen LSTMs, insbesondere BLSTMs einen größeren Kontext, wodurch eine Empfehlung dieser für feinere Granularitäten und gemischte Korpora, in denen das Kontextwissen aufgrund der verschiedenartigen Themen wichtig sein kann, ausgesprochen werden kann. Dies entspricht zum Teil den Ergebnissen aus Kapitel 5.1.2, in denen eine Empfehlung des E-CNN Klassifikators für die nachträgliche Klassifikation (Subtasks B und C) von vorher gefilterten Daten gegeben wird. Auch in diesem Fall ist der Korpus aufgrund der Filterung einheitlicher als im Fall derjenigen Subtasks, die auf dem ungefilterten Korpus operieren.

Korpus +Subtask		BLSTM		E - CNN		E - CNN - LSTM		E - LSTM - empty		E - LSTM - pre		LSTM - stacked		Durchschnitt	
		Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
BRK	A	**36.2**	**60.4**	33.5	55.5	30.3	56.5	33.0	54.4	29.1	57.8	27.7	55.7	31.6	56.7
	B	19.2	64.1	**19.3**	65.5	15.1	64.7	17.6	65.6	14.8	**71.4**	16.6	65.0	17.1	66.0
	C	**33.6**	**38.6**	23.8	36.3	21.0	35.3	21.8	36.1	19.2	33.9	21.5	35.5	23.5	36.0
	D	**33.9**	90.8	31.3	**90.9**	32.3	90.6	29.4	90.6	24.3	89.4	33.0	86.4	30.7	89.8
	E	**20.2**	38.0	19.1	38.8	17.3	39.3	18.6	37.5	16.2	37.3	18.7	**42.5**	18.3	38.9
Both	A	31.1	48.5	34.6	59.6	34.7	58.1	**37.5**	53.2	27.3	**60.0**	27.2	57.8	32.1	56.2
	B	**25.9**	54.2	24.2	55.9	23.5	**57.1**	25.6	55.7	21.1	56.8	21.9	55.5	23.7	55.9
	C	**41.8**	42.7	37.9	**44.6**	39.0	43.2	38.8	42.6	23.0	40.1	40.9	42.2	36.9	42.6
	D	34.3	90.2	**35.1**	86.4	33.5	87.1	32.8	85.8	33.2	**92.4**	29.8	87.0	33.1	88.2
	E	**25.2**	40.6	24.5	**46.3**	21.7	41.0	21.7	40.7	19.3	42.8	22.7	42.5	22.5	42.3
THF	A	35.8	**59.3**	**38.7**	52.4	36.6	58.6	34.5	51.5	36.1	52.5	26.5	51.9	34.7	54.4
	B	29.0	44.3	27.6	45.0	28.7	44.7	**29.6**	44.2	19.3	**48.0**	19.5	45.3	25.6	45.2
	C	**45.6**	54.9	41.0	**55.9**	45.4	53.1	39.9	50.8	41.4	52.5	26.7	54.5	40.0	53.6
	D	38.8	82.9	**39.7**	83.8	39.3	83.7	39.1	82.6	23.6	83.9	36.8	**85.0**	36.2	83.6
	E	**26.2**	39.6	20.6	**48.5**	21.4	45.6	21.4	46.3	21.4	48.0	14.1	48.3	20.8	46.1
Durch-schnitt		**31.8**	56.6	30.1	57.7	29.3	57.2	29.4	55.8	24.6	**57.8**	25.6	57.0	28.5	57.0

Tabelle 37: Deep-Learning Sentence-Tagging Ergebnisse.

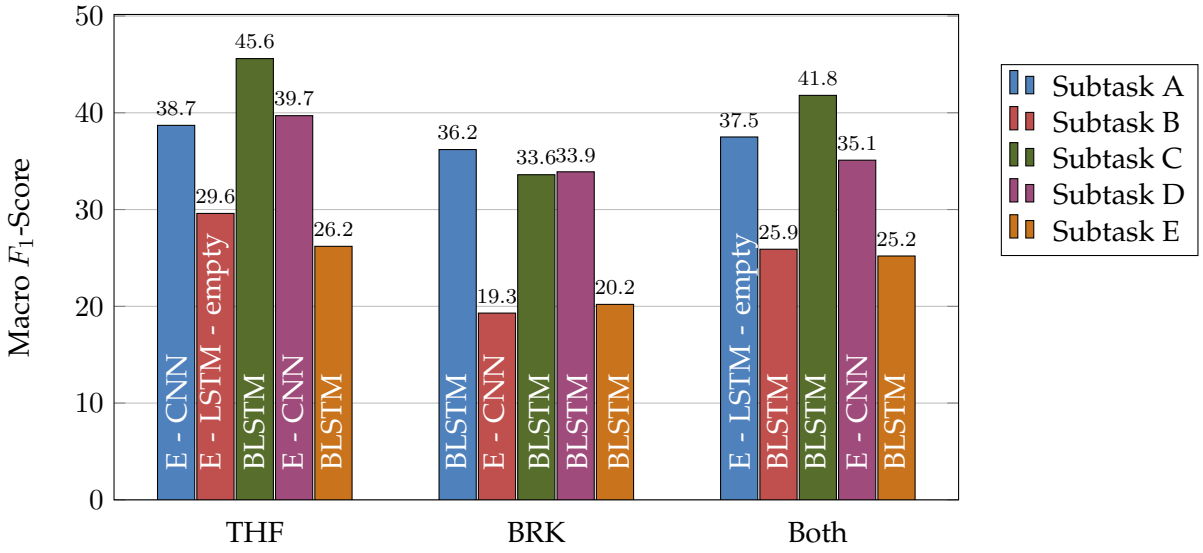


Abbildung 29: Beste Ergebnisse des Sequence-Taggings mit Deep-Learning Verfahren.

5.2.4 Klassiches Machine Learning: Cross-Platform

Die Ergebnisse des Cross-Platform Sequence Taggings fallen hinsichtlich der verwendeten Trainings- und Testsets überraschend ähnlich aus. Wird die Schwierigkeit Subtask B zu lernen sowie die große Anzahl an „O“ Labels in den Subtasks B, C und E berücksichtigt, bleiben die Ergebnisse von Subtask A und D interessant. Insbesondere das schlechtere Abschneiden von Subtask A auf Trainingsdatensatz BRK und Testdatensatz THF fällt im Gegensatz zu den Ergebnissen innerhalb der Datensätze auf (vergleiche Abbildungen 28 und 33). Dieses Verhalten unterstützt die Annahme, dass Testdatensatz THF schwierigere Klassifikationsaufgaben liefert als Testdatensatz BRK. Die durchschnittlich deutlich kürzeren Sätze im Testdatensatz THF (durchschnittlich 76.7 Tokens) gegenüber den längeren Sätzen im Testdatensatz BRK (durchschnittlich 155.2 Tokens) (siehe auch Tabelle 11) könnten hier nicht genügend Kontext liefern, um vergleichbare Resultate zum Testdatensatz BRK zu liefern. Da zur Erzeugung der Kontextsätze (siehe Kapitel 4.5.3) auch Tokens über Satzgrenzen hinweg genutzt wurden, könnte dies im Fall von inhaltlich argumentierenden Beiträgen (in denen beispielsweise Pro- und Contra-Argumente gegeneinander abgewogen werden) zu einer starken Verunreinigung des Kontexts durch inhaltlich abgegrenzte Sätze führen. Daher liegt es nahe in Zukunft den Einfluss der Art der Kontexterzeugung zu untersuchen. Zukünftige Experimente könnten die Erzeugung des Kontextes auf Satzgrenzen beschränken oder in einem Vorverarbeitungsschritt für jeden Satz einzeln hinsichtlich ausgewählter Features (z.B. grammatikalische Features, Anzahl Negationen etc.) entscheiden, ob weitere Sätze zum Kontext hinzugefügt werden

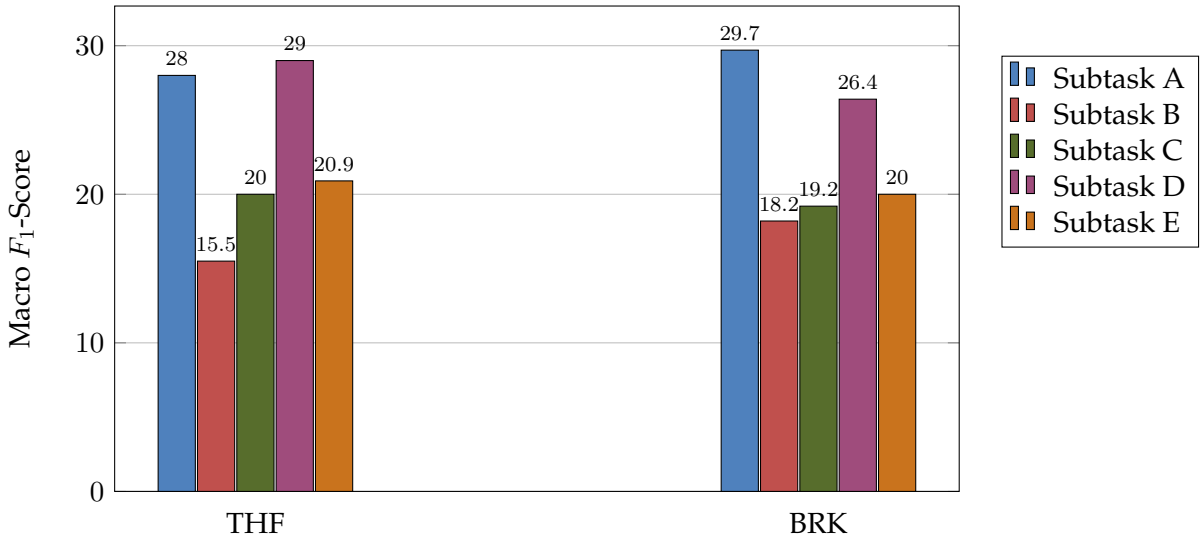


Abbildung 30: Beste Ergebnisse für **Klassifikator Random Forest**.

dürfen oder nicht.

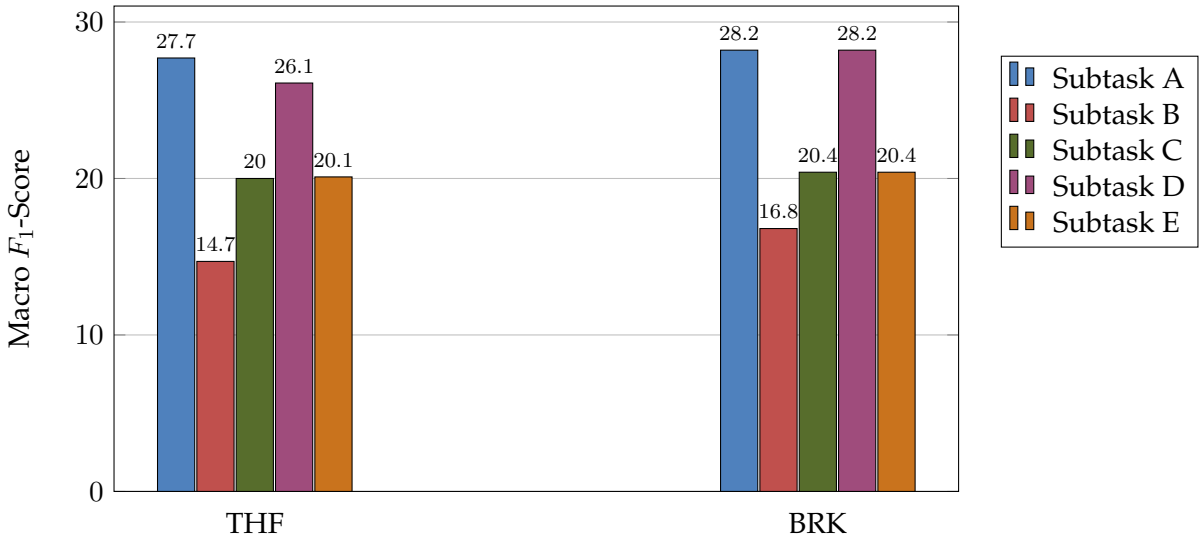


Abbildung 31: Beste Ergebnisse für **Klassifikator Support Vector Machine**.

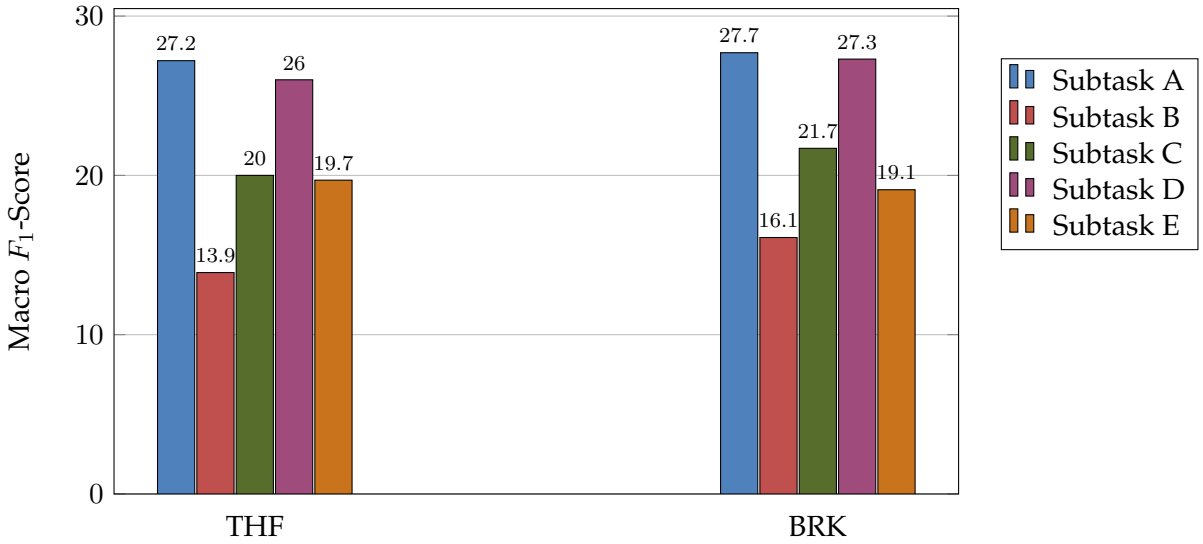


Abbildung 32: Beste Ergebnisse für **Klassifikator Support Vector Machine: Linearer Kernel**.

Klassifikator	RF		SVM		SVM-Linear		Durchschnitt	
Durchschnittlicher $F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	18.1	39.9	18.4	61.8	20.0	54.4	18.8	52.0
Grammatical ②	19.2	47.7	19.0	38.9	16.1	27.4	18.1	38.0
LDA Wikipedia ③	9.2	23.4	10.0	28.5	14.8	48.5	11.3	33.5
LDA Dataset ④	6.1	10.3	7.4	16.9	10.7	32.6	8.1	19.9
Character n-grams	20.0	57.8	19.8	50.4	17.1	29.5	19.0	45.9
Word Embeddings ⑤	18.8	46.6	19.7	49.6	17.4	33.3	18.6	43.2
Character Embeddings ⑥	19.7	46.1	21.1	47.5	19.4	36.6	20.1	43.4
①+②	18.7	41.9	20.5	59.3	20.6	54.9	19.9	52.0
①+③	16.3	38.1	18.6	57.7	19.8	54.0	18.2	49.9
①+④	19.1	44.1	20.2	58.4	20.0	54.3	19.8	52.3
①+⑤	18.4	45.6	18.5	60.3	19.8	55.0	18.9	**53.6**
①+⑥	19.8	45.5	21.4	54.7	21.2	55.8	**20.8**	52.0
①+②+⑤	18.9	42.8	20.7	60.7	20.2	55.8	19.9	53.1
①+②+⑥	19.1	43.0	21.5	54.5	21.5	55.2	20.7	50.9
①+②+⑤+⑥	18.7	38.2	21.8	57.4	21.5	55.9	20.7	50.5
Durchschnitt	17.3	40.7	18.6	**50.4**	**18.7**	46.9	18.2	46.0

Tabelle 38: Durchschnittliche F_1-Scores des Cross-Platform Sequence-Taggings.

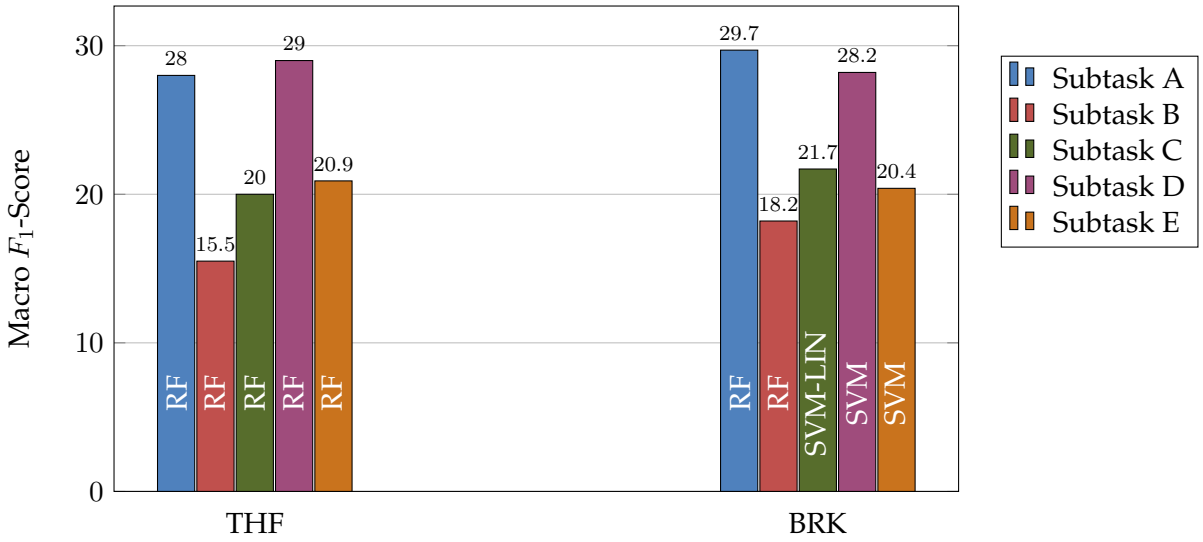


Abbildung 33: Beste Cross-Platform Ergebnisse des Sequence Taggings mit klassischen Machine Learning Verfahren.

5.2.5 Deep Learning: Cross-Platform

In diesem Kapitel werden die Cross-Platform Deep Learning Ergebnisse auf feiner Granularitätsebene (Sequence Tagging) vorgestellt. Im Vergleich zu den Resultaten des klassischen Machine Learnings fallen die Ergebnisse analog zu Kapitel 5.2.3 besser aus. Im Vergleich zu den Ergebnissen innerhalb der Datensätze können die Klassifikatoren jedoch keine guten Ergebnisse liefern. Insbesondere das gute Abschneiden von Klassifikator BLSTM für Subtask C ist nicht mehr gegeben. Subtask A und D fallen, wie zu erwarten war, besser aus als die restlichen Subtasks. Insgesamt kann sich jedoch wieder der BLSTM Klassifikator durchsetzen, wobei häufig auch die Verfahren E-CNN und E-LSTM gute Ergebnisse liefern konnten. In Kapitel 5.2.3 wurden das Verhalten des BLSTM-Klassifikators im Vergleich zum E-CNN Klassifikator bereits evaluiert. Gerade das bessere Abschneiden des E-CNN auf Testdatensatz THF (siehe Tabelle 39) unterstützt diese Annahme.

Korpus +Subtask		BLSTM		E - CNN		E - CNN - LSTM		E - LSTM - empty		E - LSTM - pre		LSTM - stacked		Durchschnitt	
		Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
BRK	A	30.4	52.7	29.4	55.9	30.0	51.4	**32.5**	50.2	28.1	**56.8**	25.0	51.1	29.2	53.0
	B	14.6	43.3	**14.7**	43.4	13.5	41.9	13.8	42.7	9.8	**44.2**	14.2	43.1	13.4	43.1
	C	20.6	34.1	**24.0**	34.4	19.5	35.5	21.5	**39.2**	17.6	30.1	17.9	30.9	20.2	34.0
	D	33.2	78.9	32.9	88.8	32.4	84.6	31.1	78.8	24.1	**89.3**	**34.3**	78.9	31.3	83.2
	E	19.0	42.5	**19.5**	43.0	17.6	39.7	18.3	39.0	16.5	**46.3**	18.4	36.8	18.2	41.2
THF	A	**33.4**	56.5	32.2	**57.9**	29.5	51.1	31.0	52.6	25.3	56.1	25.6	50.7	29.5	54.1
	B	14.8	37.1	16.3	42.6	15.1	47.5	**16.7**	43.4	13.5	**55.8**	13.9	47.1	15.1	45.6
	C	26.9	**33.1**	21.7	28.7	24.1	29.6	**28.2**	32.5	20.6	27.1	17.1	28.9	23.1	30.0
	D	30.4	91.9	**34.2**	91.4	31.2	91.5	32.5	90.5	23.8	89.6	24.3	**92.1**	29.4	91.2
	E	**26.2**	**39.7**	14.2	34.5	19.6	38.0	21.2	36.8	16.2	35.5	20.6	38.5	19.7	37.2
Durch-schnitt		**24.9**	51.0	23.9	52.1	23.2	51.1	24.7	50.6	19.6	**53.1**	21.1	49.8	22.9	51.3

Tabelle 39: Cross-Platform Deep-Learning Sentence-Tagging Ergebnisse.

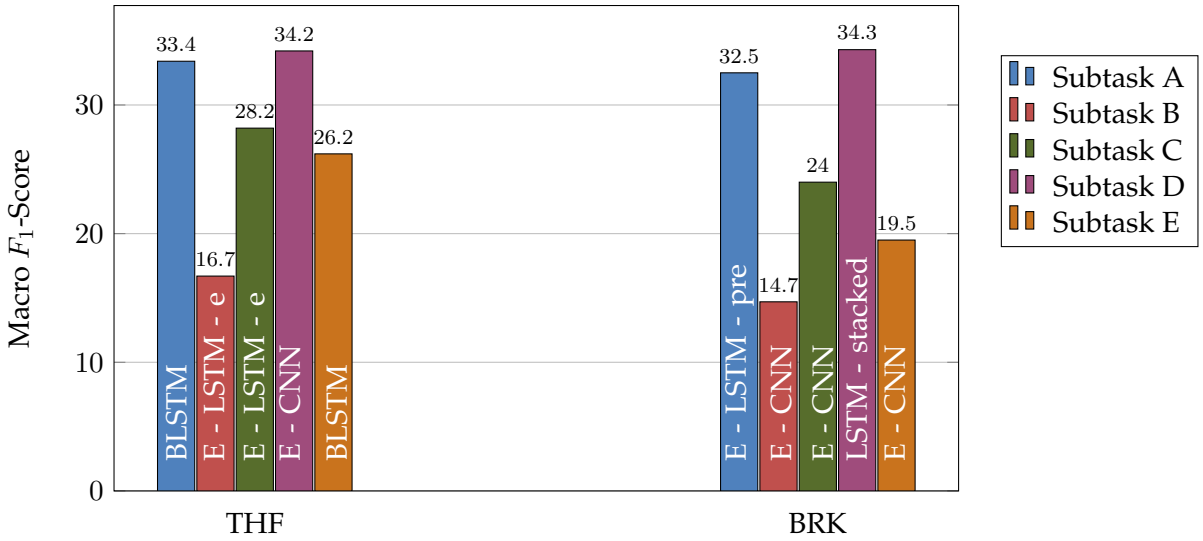


Abbildung 34: Beste Cross-Platform Ergebnisse des Sequence-Taggings mit Deep-Learning Verfahren.

5.3 Document-Tagging

In den folgenden Kapiteln werden die Ergebnisse des Document Taggings vorgestellt. Detaillierte Ergebnisse finden sich im Anhang in Kapitel A.3.

5.3.1 Klassisches Machine Learning

Detaillierte Tabellen zu den Document Tagging Ergebnissen mit klassischen Machine Learning Verfahren finden sich im Anhang in Kapitel A.3.1.

Die Ergebnisse des Document-Taggings fallen insgesamt ähnlich zu denen des Sentence-Taggings aus. Erneut zeigen sich die gleichen Tendenzen bezüglich der einzelnen Subtasks: Subtask A schneidet am besten ab, dicht gefolgt von Subtask C, D und E. Subtask B hingegen zeigt deutlich schlechtere Ergebnisse, was sich erneut mit der höheren Anzahl der Klassen (3 statt 2) sowie einem starken Klassenungleichgewicht begründen lässt. Gerade Datensatz BRK zeigt, wie auch schon im Sentence-Tagging, deutlich schlechtere Ergebnisse für Subtask B. Ein Grund dafür kann die geringe Anzahl an Vorschlägen im BRK Korpus darstellen. Insgesamt fallen beinahe alle Ergebnisse schlechter aus, als die vergleichbaren Experimente im mittel-granularen Sentence-Tagging. Dies kann sowohl an der geringeren Anzahl an verfügbaren Datenpunkten (ein Datenpunkt pro Dokument) als auch an der geringeren inhaltlichen Genauigkeit der Zusammenfassung von ganzen Dokumenten zu Datenpunkten liegen:

Liefern nur wenige Textteile in einem Dokument das Label zur Klassifikation, so kann das Training durch den dann höheren Anteil an nicht-relevanten Textteilen negativ beeinflusst werden. Die Frage bleibt hier, ob die gewählte Methode zur Vergabe der Label auf Dokumentebene unter Berücksichtigung der inhaltlichen Anforderungen eines potenziellen Nutzers des Klassifikationssystems verbessert werden könnte. In dieser Arbeit genügt das einmalige Vorkommen eines Labels in einem der Sätze eines Dokumentes, um dieses mit dem Label auf Dokumentebene zu codieren. Auf diese Weise werden auch Dokumente, die nur zu einem geringen Teil der jeweiligen Klasse entsprechen dieser Klasse zugeordnet. Gerade für längere Dokumente mit geringen relevant codierten Textteilen kann dies jedoch zu schlechten Trainingsinstanzen führen. Zusätzlich wurde für Dokumente, welche mehrere, sich gegenseitig ausschließende, Label beinhalteten, dasjenige Label gewählt, welches am häufigsten in diesem Dokument codiert wurde (Majority Voting). Bei gleichem Vorkommen wurde dasjenige Label, welches laut Codebuch zu präferieren war, verwendet, bzw. wenn keine Präferenz angegeben war, ein zufälliges Label gewählt. Durch dieses Vorgehen kann die Relevanz des codierten Labels im Vergleich zum Umfang des Dokumentes gering ausfallen.

Eine Lösung dieses Problems könnte eine Duplikation von Dokumenten mit mehreren Labeln (für jedes vorhandene Label ein Duplikat) oder eine dynamische Anpassung der Granularität sein. Im zweiten Fall würden Dokumente, bei denen die Anzahl der darin enthaltenen Klassenlabels sehr ähnlich ist, in ihre Sätze mit den jeweiligen Labels aufgeteilt werden. Eine Einschätzung der Leistung eines solchen Verfahrens wird in Kapitel

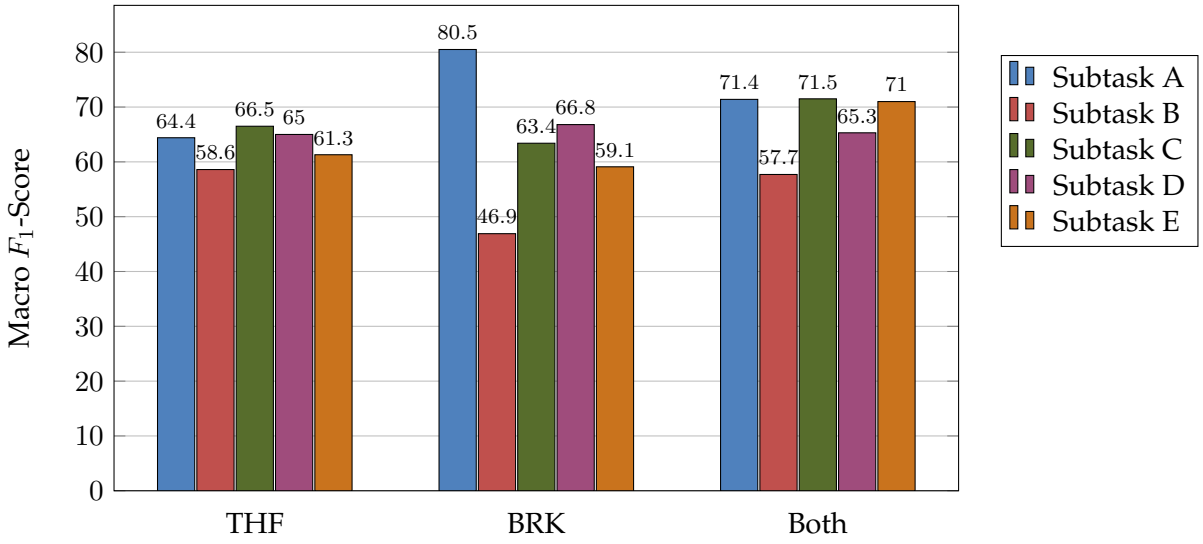


Abbildung 35: Beste Ergebnisse für den **Klassifikator Random Forest**.

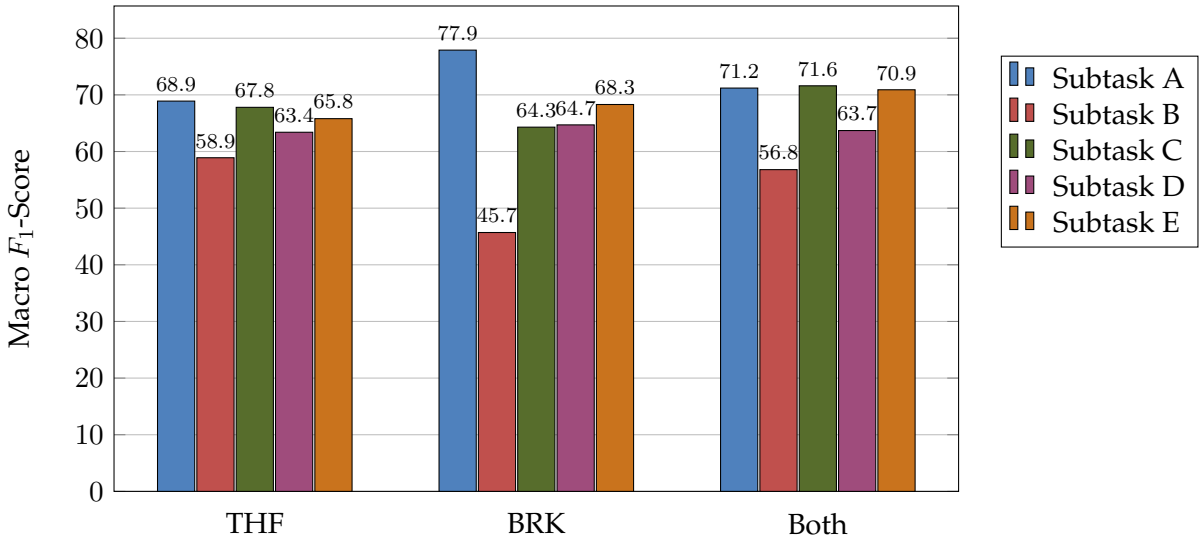


Abbildung 36: Beste Ergebnisse für den **Klassifikator Support Vector Machine**.

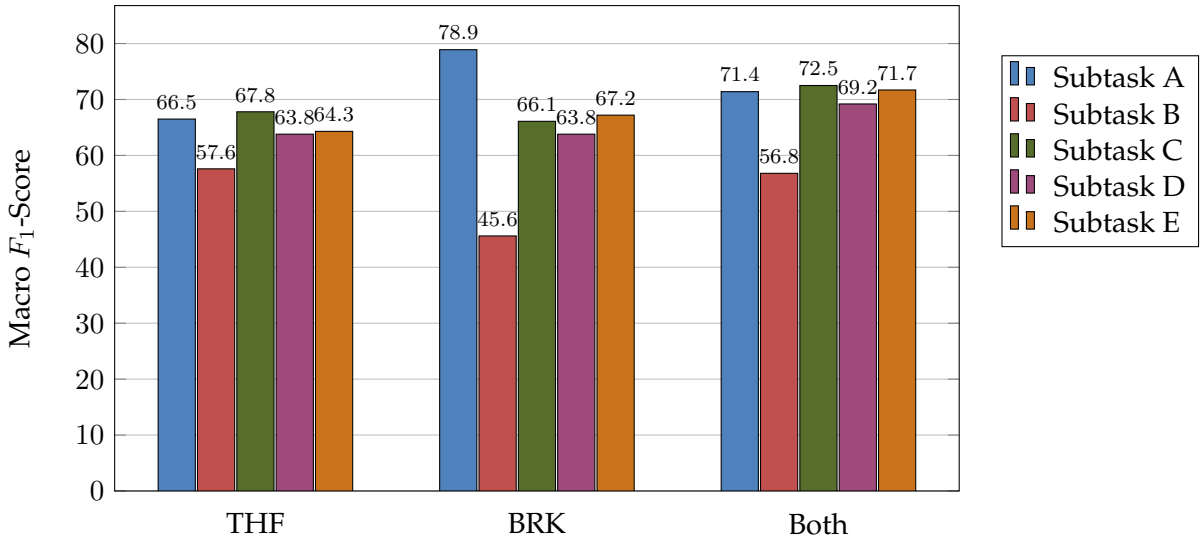


Abbildung 37: Beste Ergebnisse für den **Klassifikator Support Vector Machine: Linearer Kernel**.

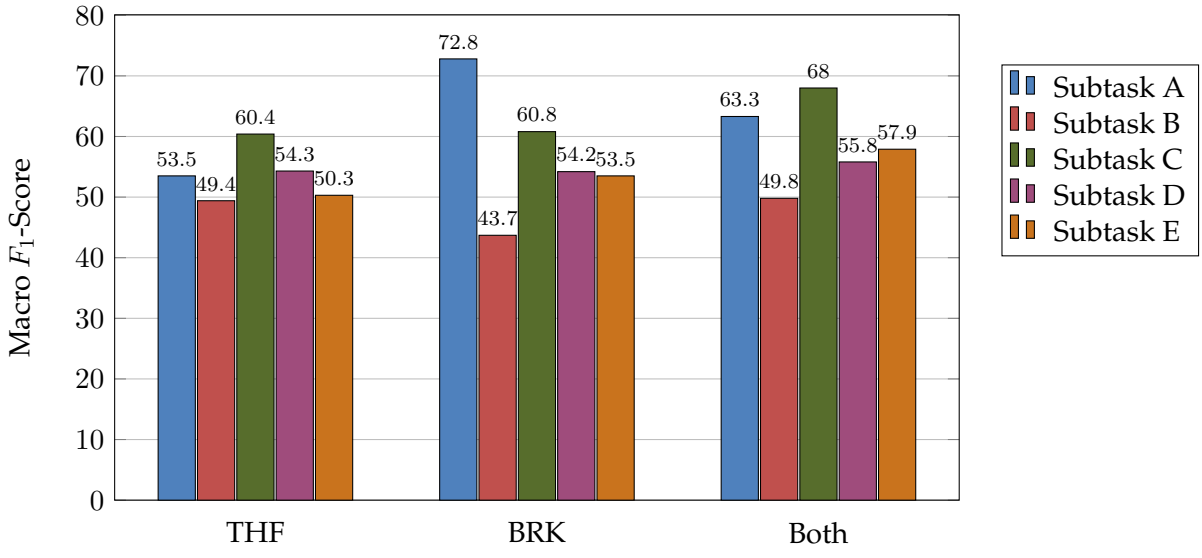


Abbildung 38: Beste Ergebnisse für den **Klassifikator CRF**.

Klassifikator	RF		SVM		SVM-Linear		CRF		Durchschnitt	
Durchschnittlicher $F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	57.3	71.5	49.8	71.7	62.4	72.4	51.0	61.8	55.1	69.3
Grammatical ②	57.9	70.9	56.6	66.2	51.5	61.1	48.9	59.7	53.7	64.5
LDA Wikipedia ③	55.2	67.3	46.7	57.3	48.6	54.7	52.4	66.0	50.7	61.3
LDA Dataset ④	56.5	68.6	52.7	62.8	47.6	52.9	52.2	65.6	52.2	62.5
Character n-grams	55.7	70.8	50.8	70.3	54.8	60.5	52.4	65.5	53.4	66.8
Word Embeddings ⑤	62.3	73.2	61.5	69.5	56.8	68.2	53.6	66.9	58.5	69.4
Character Embeddings ⑥	60.8	72.4	63.0	71.1	63.0	71.3	48.9	65.2	58.9	70.0
①+②	56.6	70.6	56.0	73.2	62.6	72.7	50.0	62.9	56.3	69.8
①+③	59.2	71.1	52.8	72.4	62.0	71.7	50.7	61.7	56.2	69.2
①+④	59.3	71.6	57.3	72.8	62.3	72.3	51.6	63.0	57.6	69.9
①+⑤	62.3	73.9	55.1	73.3	62.6	72.2	51.9	63.5	58.0	70.7
①+⑥	61.8	73.1	62.8	72.1	62.6	72.0	48.7	63.7	59.0	70.2
①+②+⑤	61.5	73.3	58.0	72.7	63.3	73.1	48.1	58.9	57.7	69.5
①+②+⑥	61.6	73.0	62.4	72.1	62.7	72.4	48.3	61.9	58.8	69.8
①+②+⑤+⑥	63.5	74.5	63.6	73.1	63.7	73.4	52.9	66.1	**60.9**	**71.8**
Durchschnitt	**59.4**	**71.7**	56.6	70.0	59.1	68.1	50.8	63.5	56.5	68.3

Tabelle 40: Durchschnittliche F_1-Scores des Document-Taggings.

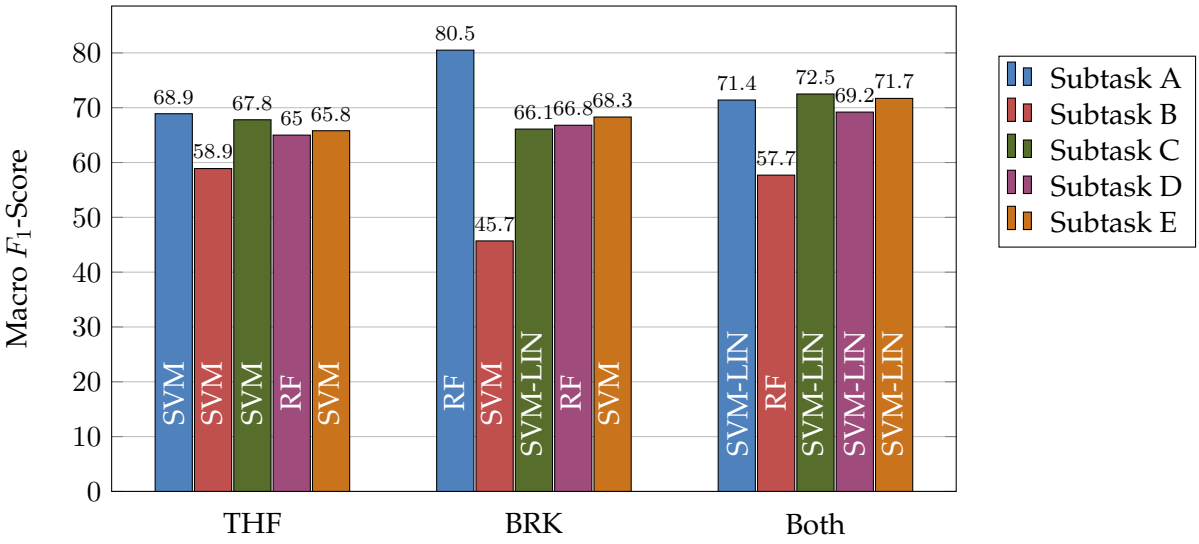


Abbildung 39: Beste Document-Tagging Ergebnisse für klassisches Machine Learning.

Klassifiziert **Gold**	argumentative	non-argumentative	Summe
argumentative	659	32	691
non-argumentative	48	80	128
Summe	707	112	819

Tabelle 41: Konfusionsmatrix für Document Tagging Klassifikator RF, Datensatz BRK, Subtask A, Features: ①+⑥.

5.4 durch die Analyse von Intergranularitätsunterschungen (siehe auch Kapitel 1.2.4) gegeben.

Eine Ausnahme der Document-Tagging Ergebnisse stellt Subtask A für Datensatz BRK dar. Hier konnten alle Klassifikatoren (insbesondere der Random Forest-Klassifikator) sehr gute Ergebnisse erzielen (Macro F_1 von 80.5) (siehe Konfusionsmatrix in Tabelle 41). Dass hierbei nicht nur die Mehrheitsklasse trainiert wurde, verwundert angesichts des starken Klassenungleichgewichtes im Trainingsset von nur 67 Instanzen der Klasse Non-argumentative und 336 Instanzen der Klasse Argumentative. Im Gegensatz zu Subtask B scheint die Aufgabe der Erkennung von argumentativen Komponenten trotz starken Klassenungleichgewichtes leichter zu sein als die Erkennung von drei Klassen, welche Ebenfalls stark ungleich gewichtet sind (Datensatz BRK, Trainingsdatensatz für Document-Tagging und Subtask B Premise: 253, Claim: 22, MajorPosition: 62).

Zusammenfassend zeigt sich das Document Tagging im Vergleich zum Sentence Tagging als schwierigere Aufgabe. Dies scheint vor allem in der Anzahl der verfügbaren Trainingsinstanzen begründet zu sein, da der kombinierte Korpus (Both) vergleichbare Ergebnisse zum Sentence Tagging liefern kann. Dies motiviert die Untersuchung von intergranularem Tagging weiter.

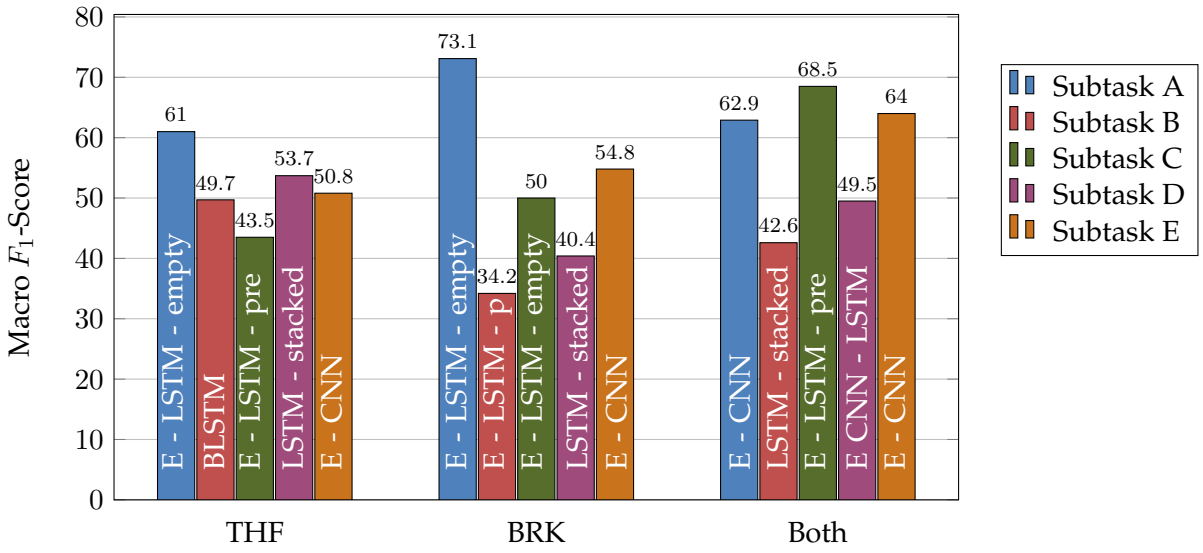


Abbildung 40: Beste Ergebnisse des Document-Taggings mit Deep-Learning Verfahren.

5.3.2 Deep Learning

Die Ergebnisse des Document-Taggings mit Deep Learning Klassifikatoren verhalten sich zu den Ergebnissen, welche mittels klassischer Machine Learning Verfahren erzielt werden konnten, analog zu den in Kapitel 5.1.2 beschriebenen Resultaten des Sentence Taggings: Insgesamt fallen die Ergebnisse schlechter aus als die vergleichbaren Ergebnisse mit klassischen Machine Learning Algorithmen und insbesondere die Datensätze BRK sowie Both zeigen deutliche Einbrüche in den Ergebnissen für Subtask D. Subtask A liefert für Datensatz BRK jedoch wieder gute Ergebnisse (vergleiche auch Kapitel 5.3.1). Erneut zeigt sich der E-CNN Klassifikator dominant und auch das, diesmal jedoch weniger ausgeprägte, bessere Abschneiden des LSTM-Netzes ohne vortrainierte Embeddings gegenüber dem LSTM-Netz mit trainierten Embeddings wiederholt sich. Es fällt auf, dass die LSTM - basierten Klassifikatoren (insbesondere E - LSTM - empty) bessere Ergebnisse auf Subtasks mit mehr Trainingsinstanzen (Subtask A und D) liefern, während Subtask E eine Stärke des Embedding CNN Klassifikators ist. Dies bestätigt die Erkenntnisse aus Kapitel 5.1.2.

Korpus +Subtask		BLSTM		E - CNN		E - CNN - LSTM		E - LSTM - empty		E - LSTM - pre		LSTM - stacked		Durchschnitt	
		Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
BRK	A	70.7	87.9	69.6	87.3	70.2	87.5	**73.1**	**88.4**	73.0	88.0	69.4	87.5	71.0	87.8
	B	34.0	80.4	29.7	80.3	31.3	80.4	34.0	80.4	**34.2**	**80.7**	31.3	80.4	32.4	80.4
	C	49.6	68.8	46.1	68.1	42.0	66.7	**50.0**	**69.5**	42.0	66.7	42.0	66.7	45.3	67.7
	D	**40.4**	**67.9**	40.4	67.9	40.4	67.9	40.4	67.9	40.4	67.9	40.4	67.9	40.4	67.9
	E	51.5	74.7	**54.8**	**76.7**	42.6	74.3	52.5	76.3	42.6	74.3	42.6	74.3	47.8	75.1
Both	A	62.8	**88.1**	**62.9**	88.0	60.2	87.3	61.1	87.3	61.8	87.7	62.1	87.2	61.8	87.6
	B	41.8	67.4	42.4	**68.3**	40.5	67.5	40.5	67.6	41.6	67.7	**42.6**	68.0	41.6	67.7
	C	64.3	65.2	65.7	66.6	66.3	68.6	58.0	64.6	**68.5**	**69.6**	67.5	68.4	65.0	67.2
	D	41.6	70.8	47.9	**71.9**	**49.5**	71.8	41.8	70.8	46.7	71.2	46.2	70.9	45.6	71.2
	E	60.9	62.9	**64.0**	**64.0**	58.0	60.1	63.1	63.3	59.0	61.1	55.0	58.3	60.0	61.6
THF	A	52.6	89.5	52.4	89.2	48.7	88.9	**61.0**	**89.8**	49.5	89.1	49.5	89.1	52.3	89.3
	B	**49.7**	60.1	43.0	**60.5**	44.1	60.2	39.8	59.2	41.5	60.0	40.3	59.8	43.1	60.0
	C	42.4	73.7	43.5	73.9	43.5	73.9	43.5	73.9	**43.5**	**73.9**	42.4	73.7	43.1	73.8
	D	42.4	72.8	46.9	**73.4**	43.5	73.1	42.4	72.8	50.6	73.3	**53.7**	73.2	46.6	73.1
	E	38.6	63.0	**50.8**	**65.3**	41.2	63.3	49.5	64.0	38.6	63.0	38.6	63.0	42.9	63.6
Durch-schnitt		49.6	72.9	**50.7**	**73.4**	48.1	72.8	50.0	73.1	48.9	72.9	48.2	72.6	49.3	72.9

Tabelle 42: Deep-Learning Document-Tagging Ergebnisse.

5.3.3 Klassisches Machine Learning: Cross-Platform

Detaillierte Tabellen zu den Cross-Platform Ergebnisse des Document-Taggings finden sich im Anhang in Kapitel A.3.2.

Eine Betrachtung der Cross-Platform Ergebnisse des Document-Taggings zeigt ähnlich zum Sentence-Tagging, dass die Ergebnisse schlechter ausfallen als für die Document-Tagging Experimente innerhalb der Datensätze. Auffällig ist jedoch im Vergleich zu den Cross-Platform Ergebnissen des Sentence-Taggings, dass diesmal ein Training auf Datensatz THF und Test auf Datensatz BRK bessere Ergebnisse für Subtask A liefert als umgekehrt (Tabelle 45). Dieses Verhalten ist genau gegenläufig zu den Beobachtungen des Sentence-Taggings aus Tabelle 23. Hier konnte ein Training auf Datensatz BRK bessere Ergebnisse für Subtask A als andersherum liefern. Das bessere Training auf Datensatz THF könnte unter Berücksichtigung von Tabelle 10 jedoch mit der deutlich höheren Anzahl an relevanten Beiträgen in diesem Korpus (1445 argumentative Beiträge in THF gegenüber 1027 argumentativen Beiträgen in Korpus BRK) begründet werden. Dies jedoch widerspricht den guten Ergebnissen für Datensatz BRK innerhalb des Korpus in Abbildung 39. Aus diesem Grund ist es wahrscheinlicher, dass der Testdatensatz THF im Fall des Document Taggings anspruchsvoller zu klassifizieren ist als der Testdatensatz BRK. Wird hierzu Tabelle 11 betrachtet, könnte ein Zusammenhang mit den durchschnittlich kürzeren Beitrags- und Satzlängen im Testdatensatz THF gegenüber BRK vermutet werden. Diese Vermutung bedarf jedoch weiterer Untersuchungen, welche in Form von zukünftigen Experimenten durch Beschränkung der Beitrags- und Satzlängen mittels Verfahren der automatisierten Textzusammenfassung (siehe z.B. (Yogan et al., 2016)) erfolgen könnte.

Bezüglich der Features liefert die Kombination ①+②+⑤+⑥ im Schnitt abermals die besten Ergebnisse. Häufig kann allerdins auch die alleinige Verwendung von Feature ⑥ (Character Embeddings) sehr gute Resultate liefern. Dabei ist anzumerken, dass sich in den meisten Fällen die auf dem Common Crawl-Corpus trainierten FastText Repräsentationen mit den besten Ergebnissen durchsetzen konnten.

Hinsichtlich der Klassifikatoren wiederholen sich die Ergebnisse, welche bereits vorher in den Kapiteln 5.1.1, 5.1.3 und 5.3.1 beobachtet wurden. Hierbei liefern der Random Forest Klassifikator sowie die beiden untersuchten SVM Varianten ähnlich gute Ergebnisse, während die Conditional Random Fields deutlich schlechter ausfallen. Unter Berücksichtigung der Ergebnisse aus Kapitel 5.2.2 sowie 5.3.1, kann davon ausgegangen werden, dass der Random Forest Klassifikator einer SVM für den Trainingsdatensatz BRK und Subtask A vorzuziehen ist. Allgemeiner kann gefolgert werden, dass sich ein Random Forest vor allem für Korpora mit längeren bzw. komplexeren Satzstrukturen, längeren Beiträgen und für Klassifikationsaufgaben mit wenigen Klassen eignet, selbst wenn ein Klassenungleichgewicht existiert.

Zusammenfassend fallen die Cross-Platform Untersuchungen auf Dokumentebene schlechter aus als die vergleichbaren Untersuchungen auf Satzebene. Besonders zu beachten ist der starke Leistungseinbruch auf Trainingsdatensatz BRK für Subtask A. Aufgrund der weiteren Ergebnisse konnte hier jedoch auf eine höhere Schwierigkeit des Testdatensatzes THF geschlossen werden und es werden weitere Untersuchungen in Abhängigkeit der durchschnittlichen Beitrags- und Satzlängen vorgeschlagen. Zusätzlich konn-

ten unter der Berücksichtigung der vorherigen Ergebnisse Empfehlungen für den Random Forest Klassifikator gegeben werden.

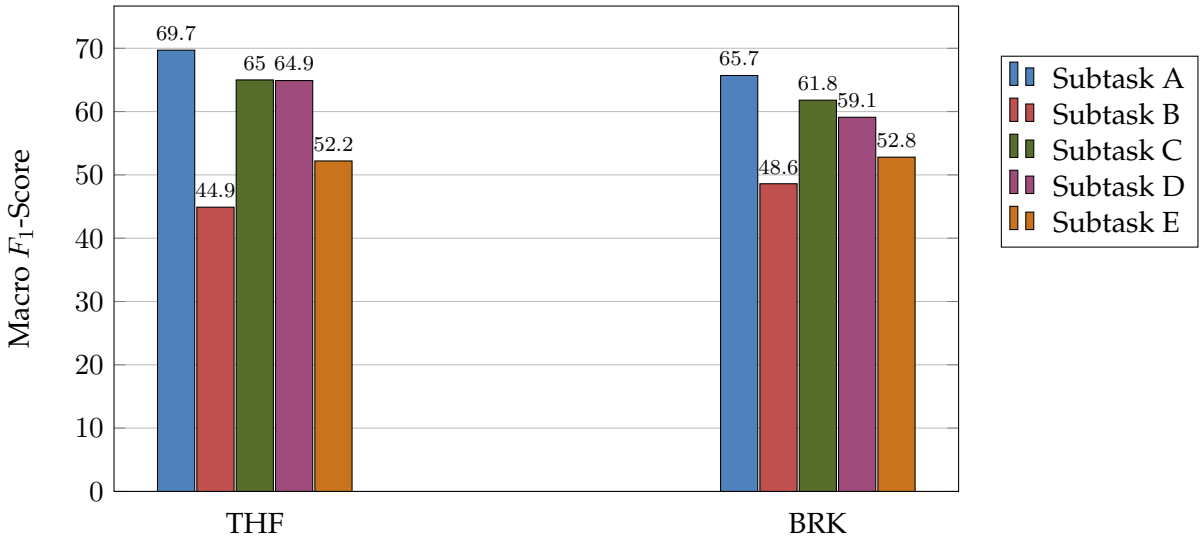


Abbildung 41: Beste Cross-Platform Ergebnisse für den **Klassifikator Random Forest**.

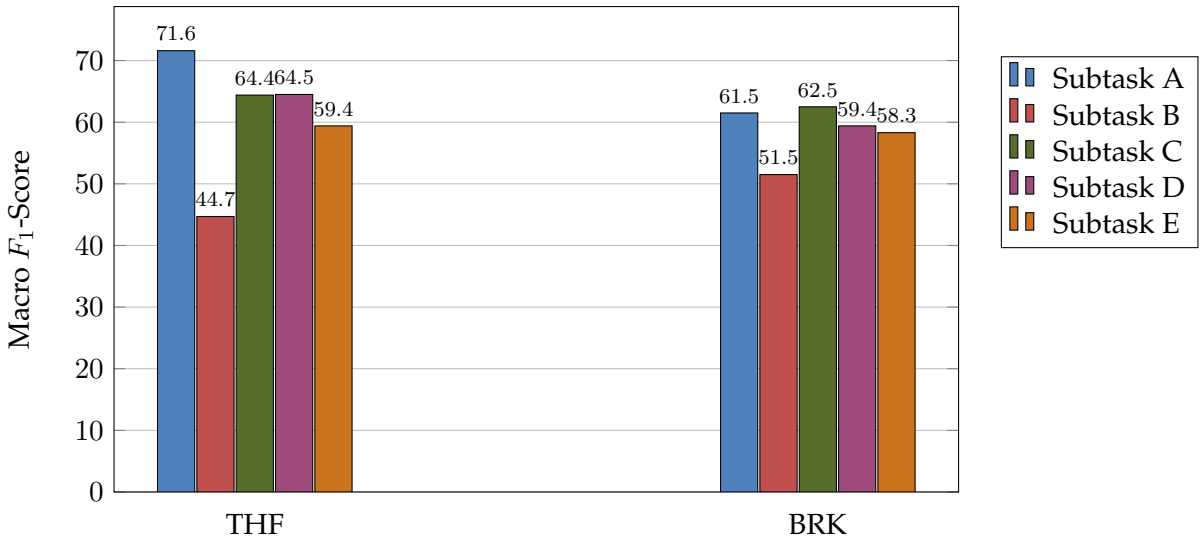


Abbildung 42: Beste Cross-Platform Ergebnisse für den **Klassifikator Support Vector Machine**.

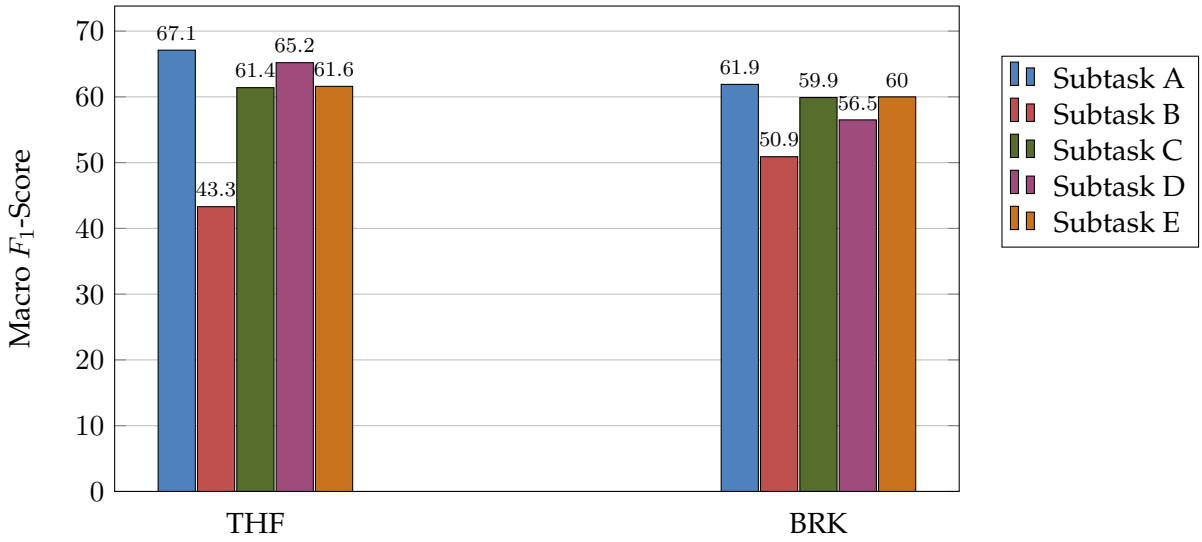


Abbildung 43: Beste Cross-Platform Ergebnisse für den **Klassifikator Support Vector Machine: Linearer Kernel**.

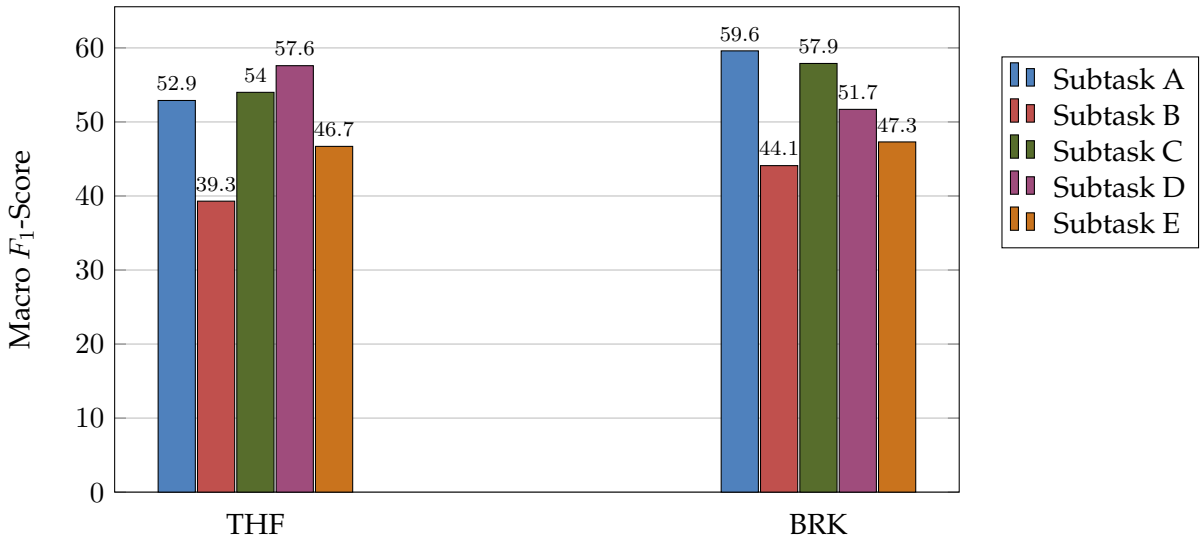


Abbildung 44: Beste Cross-Platform Ergebnisse für den **Klassifikator CRF**.

Klassifikator	RF		SVM		SVM-Linear		CRF		Durchschnitt	
Durchschnittlicher $F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	48.2	63.4	43.8	62.3	54.1	64.2	45.1	55.8	47.8	61.4
Grammatical ②	52.7	63.9	50.4	59.5	43.5	54.3	42.6	52.0	47.3	57.4
LDA Wikipedia ③	49.1	59.4	41.0	50.5	40.6	46.4	46.2	57.8	44.2	53.5
LDA Dataset ④	49.2	59.9	46.8	58.0	41.8	52.2	45.3	55.7	45.8	56.5
Character n-grams	46.8	60.6	45.4	59.1	50.1	57.7	47.6	59.0	47.5	59.1
Word Embeddings ⑤	53.0	65.9	56.8	66.5	51.6	62.1	45.6	57.5	51.8	63.0
Character Embeddings ⑥	52.3	64.8	57.1	64.7	54.3	64.7	42.2	55.2	51.5	62.3
①+②	48.0	64.0	48.7	63.3	52.2	62.7	43.0	55.0	48.0	61.2
①+③	52.1	63.5	45.3	60.4	53.8	63.9	45.6	56.0	49.2	61.0
①+④	52.6	65.4	48.9	62.9	54.2	64.2	45.0	55.8	50.2	62.1
①+⑤	52.6	66.6	47.8	63.9	53.6	64.0	44.1	55.1	49.5	62.4
①+⑥	53.0	65.9	54.8	65.6	54.5	64.3	41.0	52.3	50.8	62.0
①+②+⑤	53.8	67.8	50.2	64.0	52.9	63.7	41.9	50.9	49.7	61.6
①+②+⑥	54.0	66.9	53.7	64.7	53.6	64.5	44.4	56.6	51.4	63.2
①+②+⑤+⑥	54.8	68.1	54.4	66.3	54.0	63.5	48.9	61.0	**53.0**	**64.7**
Durchschnitt	**51.5**	**64.4**	49.7	62.1	51.0	60.8	44.6	55.7	49.2	60.8

Tabelle 43: Durchschnittliche F_1-Scores des Cross-Platform Document-Taggings.

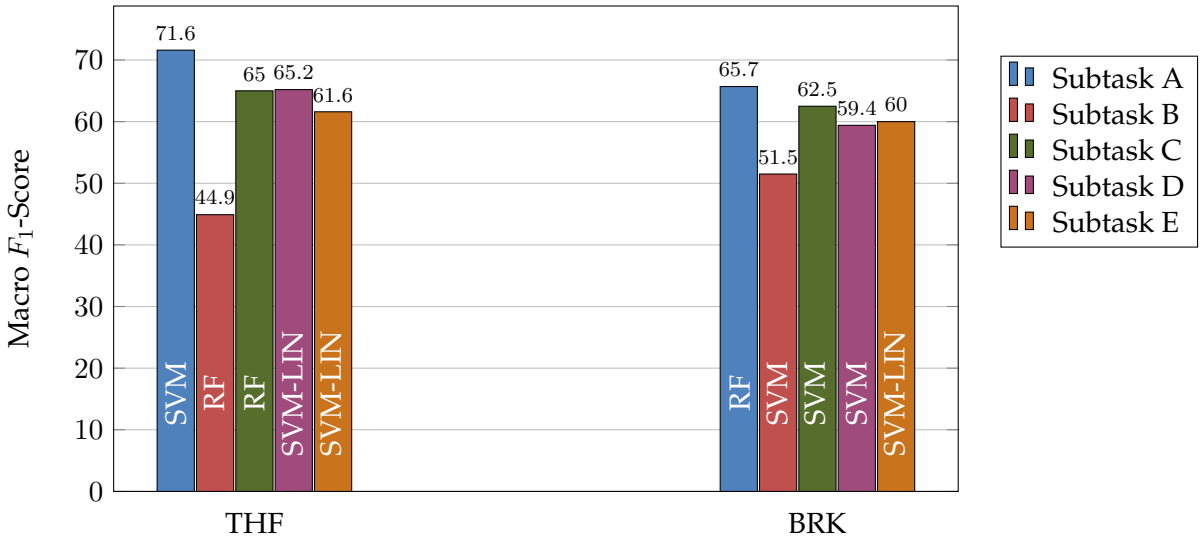


Abbildung 45: Beste Document-Tagging Ergebnisse für klassisches Machine Learning mit Cross-Platform Validation.

5.3.4 Deep Learning: Cross-Platform

Die Deep Learning Ergebnisse des Cross-Platform Document Taggings weisen erneut schlechtere Ergebnisse als die klassischen Machine Learning Verfahren auf. Hierbei konnte der Klassifikator E - LSMT- empty im Mittel die besten Resultate erzielen, wobei sich abermals zeigt, dass die E - LSTM Variante ohne vortrainierte Embeddings besser abschneidet als mit diesen. Zusätzlich ist die Leistung des BLSTM Klassifikators mit einem Training auf Datensatz THF und Test auf Datensatz BRK hervorzuheben. Hier konnte der Klassifikator vor allem mit einem Macro F_1-Score von 62.2% für Subtask A sowie den besten Werten auch für Subtask B und C auffällig gut abschneiden (siehe Tabelle 44). Da insbesondere Subtask B ein großes Klassenungleichgewicht im Trainingsdatensatz BRK aufweist, lässt dies unter Berücksichtigung der durchweg guten Ergebnisse des Klassifikators auf Datensatz THF mit Subtask B (siehe auch Tabelle 42) vermuten, dass gerade die BLSTM-Architektur mehr Trainingsinstanzen für unterrepräsentierte Klassen benötigt. Ein Grund für dieses Verhalten könnte die, der Architektur eigene, Berücksichtigung des Kontexts sein, welcher bei wenigen Trainingsinstanzen zu spezifisch gewesen sein könnte. Hinsichtlich der Subtasks ist analog zu Kapitel 5.3.3 ein besseres Abschneiden von Trainingsgrundlage THF für Subtask A zu erkennen. Subtask C verhält sich jedoch wieder wie im Fall des Sentence Taggings besser auf Trainingsdatensatz BRK als auf Trainingsdatensatz THF. Eine Erläuterung dazu findet sich in Kapitel 5.1.4.

Korpus +Subtask		BLSTM		E - CNN		E - CNN - LSTM		E - LSTM - empty		E - LSTM - pre		LSTM - stacked		Durchschnitt	
		Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
BRK	A	47.0	88.8	**56.2**	85.6	47.0	88.8	53.3	85.8	47.9	**88.9**	47.0	88.8	49.7	87.8
	B	**41.8**	59.1	23.8	55.6	39.2	59.3	39.6	**59.8**	39.9	59.5	40.1	59.8	37.4	58.9
	C	55.3	62.5	42.4	**73.7**	51.9	61.9	**58.4**	64.1	30.1	32.5	30.5	32.8	44.8	54.6
	D	42.1	72.7	42.1	72.7	**42.4**	**72.8**	42.1	72.7	42.1	72.7	42.1	72.7	42.1	72.7
	E	36.2	40.9	49.8	50.0	29.9	38.6	**51.3**	**51.3**	27.0	37.0	27.0	37.0	36.9	42.5
THF	A	**62.6**	**85.7**	54.1	85.1	45.8	84.4	48.9	84.7	53.7	85.3	45.8	84.4	51.8	84.9
	B	**34.0**	**80.4**	31.3	80.4	33.7	80.1	33.6	80.0	34.0	80.4	29.7	80.3	32.7	80.3
	C	**46.1**	**46.1**	44.6	44.7	30.3	36.2	43.9	44.0	29.6	36.2	25.4	34.0	36.6	40.2
	D	41.2	68.1	44.6	68.9	42.0	68.1	**57.0**	**70.1**	48.4	68.5	48.6	68.7	47.0	68.7
	E	45.8	49.4	**47.4**	**60.9**	42.2	44.3	44.0	44.7	20.4	25.7	20.4	25.7	36.7	41.8
	Durch-schnitt	45.2	65.4	43.6	**67.8**	40.4	63.5	**47.2**	65.7	37.3	58.7	35.7	58.4	41.6	63.2

Tabelle 44: Cross-Platform Deep-Learning Document-Tagging Ergebnisse.

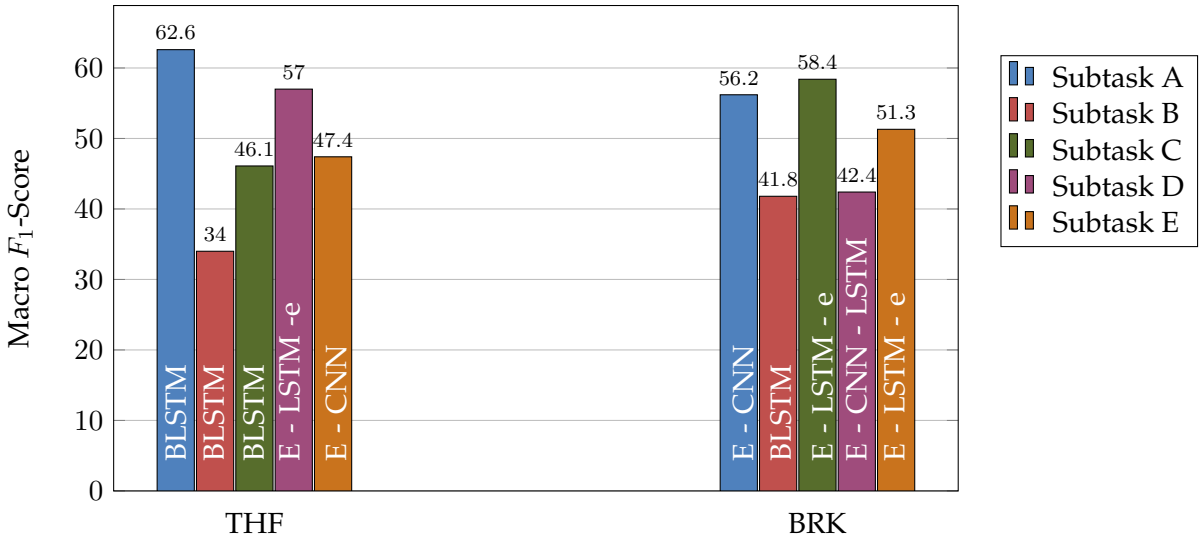


Abbildung 46: Beste Cross-Platform Ergebnisse des Document-Taggings mit Deep-Learning Verfahren.

5.4 Intergranularität

In diesem Kapitel folgen die Ergebnisse der durch die vorherigen Untersuchungen motivierten intergranularen Experimente (siehe Kapitel 1.2.4). Hierbei werden die in Kapitel 4.5.4 vorgestellten Verfahren für intergranulares Tagging verwendet.

Im Vergleich der Ergebnisse des Document Taggings, welche mit den klassischen Machine Learning Verfahren auf Datensatz THF und Subtask B erreicht werden konnten, mit jenen, die durch intergranulares Training (IG Training) oder intergranularem Post-Processing (IG PostProc) erzielt werden konnten (Abbildung 47), können klare Tendenzen erkannt werden. Einerseits kann der Klassifikator RF mit beiden Intergranularitätsmethoden besser abschneiden als das normale Document Tagging, auf der anderen Seite schneidet die Post-Processing Variante für beide SVM-Varianten höchsten genauso gut oder schlechter als das normale Document Tagging ab. Eine zusätzliche Untersuchung in dem die Ergebnisse der drei untersuchten Klassifikatoren im Sinne eines Ensemble-Klassifikators kombiniert wurden (Sammlung der Labels aller Klassifikatoren für intergranulares Post-Processing) lieferte dabei nur einen Macro F_1-Score von 52.2% und schnitt damit schlechter ab als die Einzelergebnisse. Intergranulares Training hingegen liefert in allen Fällen geringfügig bessere Resultate. Werden zusätzlich die folgenden Resultate des intergranularen Sentence Taggings in Abbildung 49 berücksichtigt, so kann gefolgert werden, dass intergranulares Training dem intergranularen Post-Processing vorzuziehen ist. Dieser Umstand kann insbesondere für das intergranulare Sentence Tagging, welches auf den Daten des Sequence Taggings trainiert wird, dadurch begründet sein, dass in den hier vorgestellten Experimenten mit intergranularem Training das Klassifikationsschema Default verwendet wurde, was die Anzahl der Klassen im Vergleich zum üblichen BIO-Schema deutlich reduziert. Aufgrund der Ergebnisse aus Tabelle 33 kann jedoch davon ausgegangen werden, dass die Nutzung des BIO-Schemas für das Training auf Sequence Ebene mit anschließender Konvertierung der Labels auf die Default-Klassen nach der Klassifikation ebenfalls gute Ergebnisse liefern könnte. Ein so trainierter Klassifikator kann dann auch für reines Sequence Tagging verwendet werden, da der Klassifikator auf den korrekten BIO-Labels trainiert wurde.

Werden der umfangreicheren Vergleich der besten Resultate für alle Subtasks mit den Ergebnissen des intergranularen Post-Processings in Abbildung 48 untersucht, so zeigen sich klare Muster in den Ergebnissen. Gerade jene Subtasks mit wenigen Trainingsinstanzen (Subtasks C und E) können von intergranularem Training profitieren, während die restlichen Subtasks meist schlechter abschneiden. Dies entspricht der intuitiven Annahme, dass eine Erhöhung der Trainingsinstanzen durch Intergranularität immer mit einer Verschlechterung der Güte jeder einzelnen Trainingsinstanz einhergeht und somit nur für Randfälle, in denen sehr wenige Datenpunkte vorliegen, sinnvoll ist. In diesem Sinne kann eine Empfehlung für intergranulare Klassifikation in den gleichen Fällen ausgesprochen werden, in denen auch klassische Oversampling-Methoden (siehe Kapitel 4.6) Anwendung finden: Klassifikationsaufgaben mit wenigen Trainingsinstanzen. Die Funktion des Ausgleichs eines Klassenungleichgewichtes, welche durch Oversampling-Verfahren ebenfalls häufig geliefert wird, kann für die Intergranularitätsverfahren nur bedingt bestätigt werden, da insbesondere Subtask B in Abbildung 48 deutlich schlechter ausfällt als die normalen Ergebnisse. Dennoch ist eine zukünftige Untersuchung von intergranularem Training empfehlenswert. Insbesondere die leicht verbesserten Ergebnisse

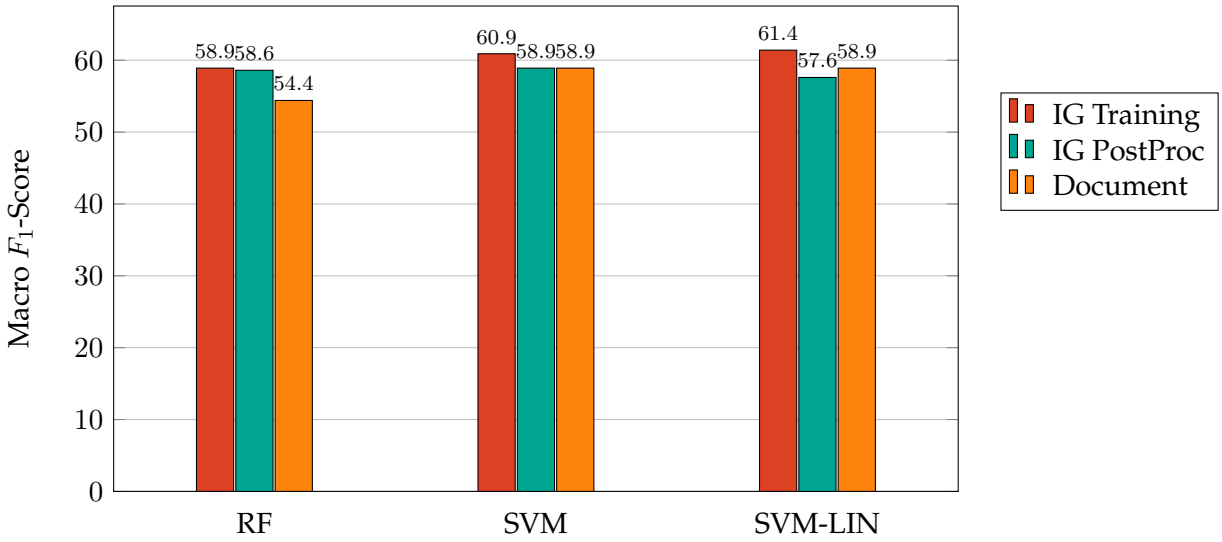


Abbildung 47: Vergleich der besten Resultate für Datensatz THF und Subtask B. Es werden die besten Ergebnisse des Document Taggings mit den besten Ergebnissen der Klassifikation mit intergranularem Training verglichen.

des Document Taggings in Abbildung 47 und 48 motivieren eine weitere Untersuchung.

Durch Betrachtung der Ergebnisse von intergranularer Klassifikation für die Anwendung des Sentence Taggings (Abbildung 49) zeigen sich bessere Ergebnisse für den schwierigeren Subtask B auf Datensatz THF (drei Klassen, leichtes Klassenungleichgewicht) und wieder deutlich schlechtere Ergebnisse für intergranulares Post-Processing im Vergleich zum intergranularen Training.

Eine Verwendung des gut abschneidenden Sequence Tagging Ergebnisses des BLSTM Klassifikators für Subtask C auf Datensatz THF (vergleiche Kapitel 5.2.3) für intergranulares Post-Processing erzielte nur einen Macro F_1-Score von 48.1%, was im Vergleich zum besten Sentence Tagging Ergebnise (70.9%) deutlich schlechter ausfällt. Eine Untersuchung von intergranularem Training für Deep Learning Klassifikatoren konnte aufgrund von zeitlichen Gründen nicht mehr in dieser Arbeit vorgenommen werden und wird aufgrund der verhältnismäßig guten Resultate dieser Verfahren in Kapitel 5.2.3 empfohlen.

Zusammenfassend liefern die Methoden der intergranularen Klassifikation nur in speziellen Fällen Verbesserungen der Ergebnisse. Eine Empfehlung der Verwendung ist daher vor allem für Randfälle mit sehr wenigen Trainingsinstanzen auszusprechen. Ist die Möglichkeit von intergranularem Training bereits aufgrund der Untersuchung von mehreren Granularitätsebenen gegeben, so bietet sich eine Untersuchung der Ergebnisse von intergranularem Training für Klassifikationsaufgaben mit wenigen Trainingsinstanzen an und kann als Alternative oder in Verbindung mit Oversampling-Verfahren genutzt werden.

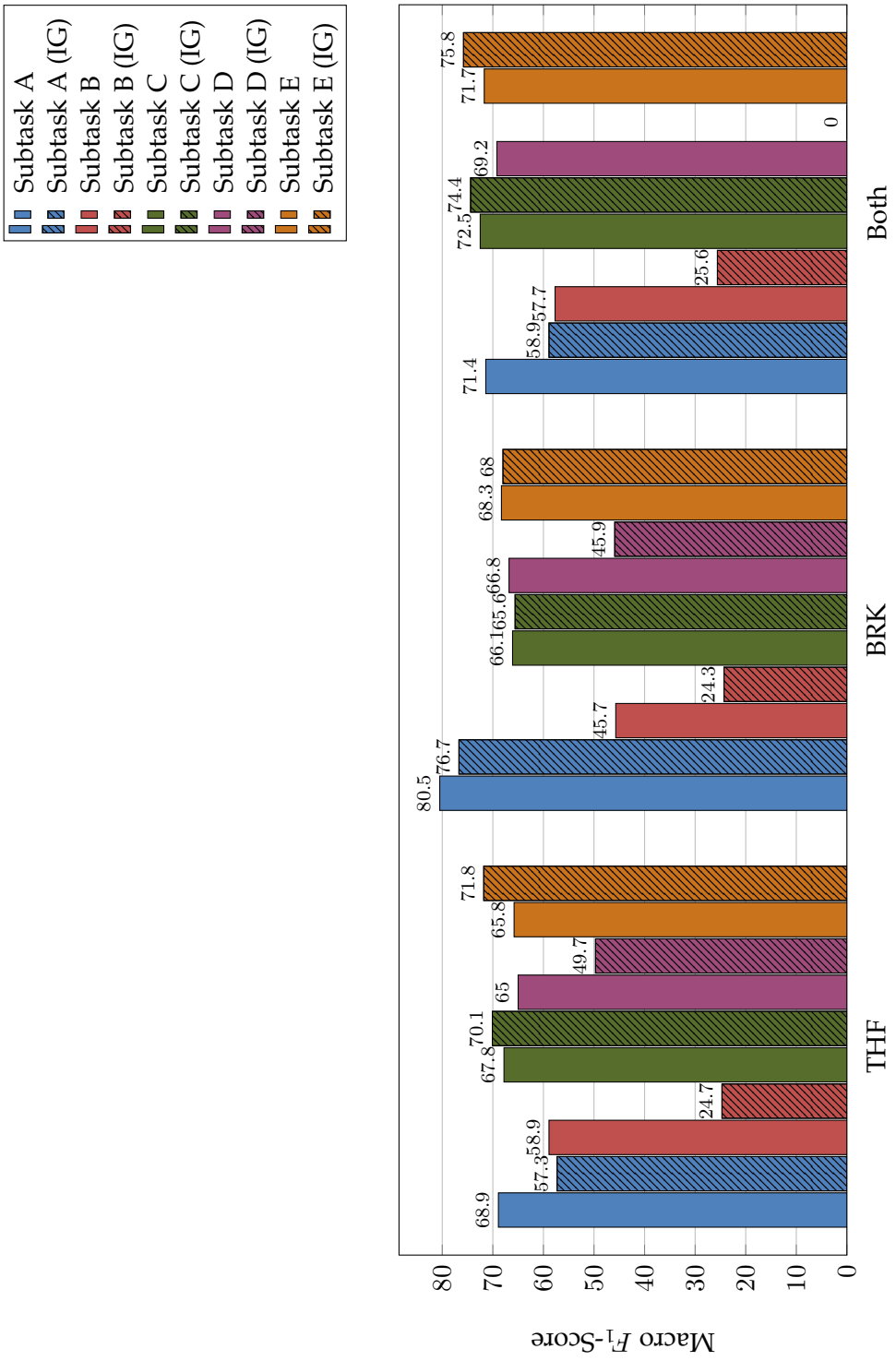


Abbildung 48: Vergleich der besten Document Tagging Ergebnisse mit den Ergebnissen der Klassifikation mit intergranularem Post Processing. Die intergranularen Ergebnisse sind schraffiert dargestellt und in der Legende mit IG bezeichnet.

Klassifikator	RF		SVM		SVM-Linear		Durchschnitt	
Durchschnittlicher $F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	**57.0**	**66.9**	-	-	-	-	57.0	**66.9**
Grammatical ②	-	-	-	-	**52.2**	**58.3**	52.2	58.3
LDA Wikipedia ③	42.8	**53.5**	-	-	**43.3**	50.2	43.0	51.9
LDA Dataset ④	**44.1**	**51.6**	37.3	46.7	-	-	40.7	49.2
Character n-grams	**51.5**	**61.1**	42.3	41.1	-	-	46.9	51.1
Word Embeddings ⑤	54.0	63.2	-	-	**57.9**	**64.8**	56.0	64.0
Character Embeddings ⑥	52.6	**63.5**	**57.0**	61.7	56.4	62.0	55.3	62.4
①+②	53.7	**66.0**	-	-	**61.0**	65.4	57.4	65.7
①+③	**57.9**	**66.5**	-	-	-	-	57.9	66.5
①+④	-	-	50.9	**64.4**	**52.9**	60.5	51.9	62.5
①+⑤	-	-	35.8	59.1	**58.5**	**65.6**	47.1	62.3
①+⑥	56.7	65.3	60.2	64.1	**61.0**	**67.3**	59.3	65.6
①+②+⑤	-	-	50.6	64.5	**61.4**	**65.4**	56.0	65.0
①+②+⑥	56.5	**66.5**	57.3	60.3	**59.0**	62.4	57.6	63.1
①+②+⑤+⑥	**58.9**	**67.3**	57.0	59.8	58.7	61.2	**58.2**	62.8
Durchschnitt	51.7	62.5	53.8	58.8	**58.8**	**63.2**	53.1	61.2

Tabelle 45: Durchschnittliche F_1-Scores des Document Taggings mit intergranularem Post-Processing.

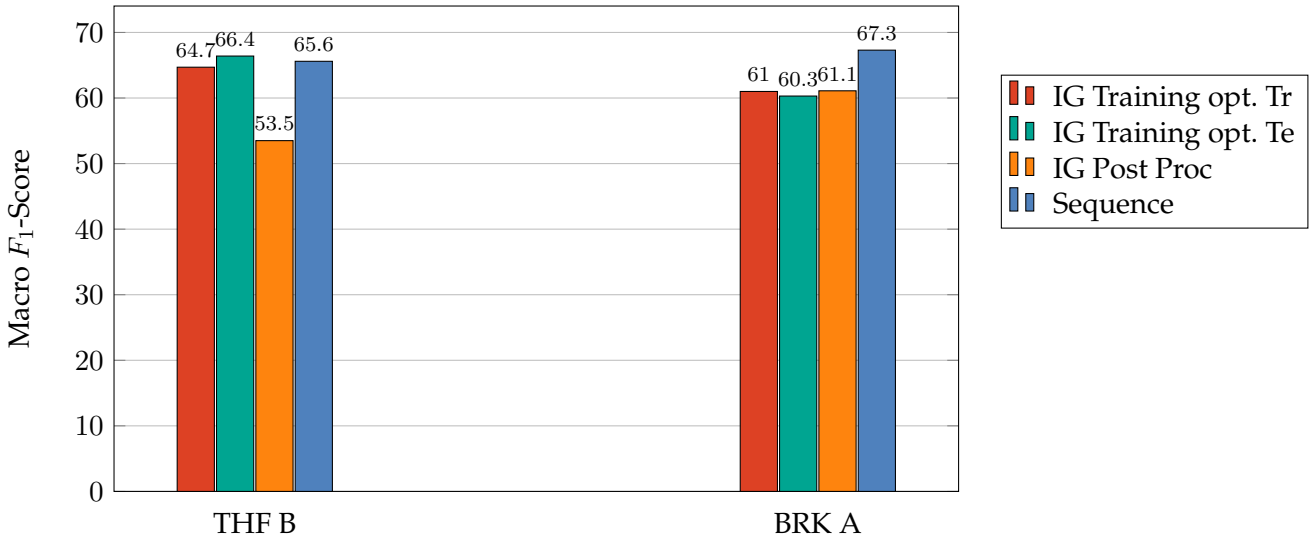


Abbildung 49: Vergleich der besten Sentence Tagging Resultate für Datensatz THF und Subtask B und BRK A. Es werden die besten Ergebnisse des Sentence Taggings mit den besten Ergebnissen der Klassifikation mit intergranularem Training (Trainingsgranularität: Sequence Tagging), welches einmal mit den optimalen Parametern für die Granularität des Trainingssets und einmal mit den optimalen Parametern für die Granularität des Testsets untersucht wurde (bezeichnet mit IG Training opt. Tr bzw. Te), verglichen.

6 Fazit

Es folgte eine Zusammenfassung der Ergebnisse sowie ein Ausblick für zukünftige Arbeiten.

6.1 Zusammenfassung

In dieser Arbeit wurde die Verwendung von Machine Learning Algorithmen zur Klassifikation von Daten aus Online-Partizipationsplattformen motiviert und experimentell untersucht. Dabei wurden zwei, im Zuge von Katharina Esaus Doktorarbeit (siehe auch Esau (2018)), codierte Korpora zu den Themen Braunkohleabbau in NRW (BRK) und Nutzung eines ehemaligen Flughafengeländes in Berlin (THF) verwendet. Es wurden für die Betreiber dieser Plattformen potenziell relevante Klassifikationsaufgaben im Bereich des Argument Minings (Subtasks A, B und C) sowie Sentiment Analysis (Subtasks D und E) vorgestellt und diese bezüglich drei verschiedener Granularitäsebenen spezifiziert. Es wurden Methoden des klassischen Machine Learnings, im Speziellen die Klassifikatoren Random Forest, Support Vector Machines mit linearem und RBF Kernel sowie Conditional Random Fields, und Deep Learning Verfahren auf Basis von LSTMs und Convolutional Neural Nets beschrieben und eine Übersicht über mögliche Features geliefert. Hierbei wurden einfache Features wie Unigram-Verteilungen aber auch grammatische Features und Word- sowie Character Embeddings sowie deren Kombinationen inklusive Optimierung der Parameter durch Gridsearch verwendet. Zusätzlich wurden

Oversampling-Verfahren beschrieben und zwei Verfahren der Intergranularitätsuntersuchungen als eigene Idee zur Verbesserung der Klassifikationsergebnisse vorgeschlagen. Es folgten dann umfangreiche Experimente auf den drei eingeführten Granularitätsebenen und den damit korrespondierenden Aufgaben des Sentence Taggings, Document Taggings sowie Sequence Taggings, welche in Kapitel 5 ausgewertet wurden. Zusätzlich wurden Cross-Platform Experimente durchgeführt, bei denen jeweils ein Korpus als Quelle der Trainingsgrundlage und der jeweils andere Korpus als Quelle des Testdatensatzes diente. Experimente zum Einfluss von Oversampling sowie intergranulare Untersuchungen wurden ebenfalls durchgeführt. Alle Ergebnisse wurden analysiert und mit Beispielen aus den Datensätzen illustriert. Zusätzlich wurden detaillierte Ergebnisstabellen zu allen Experimenten im Anhang zur Verfügung gestellt.

Zusammenfassend konnten die folgenden Aussagen getroffen und Empfehlungen ausgeprochen werden: Insgesamt konnten die Untersuchungen auf mittlerer Granularitätsebenen die besten Resultate aufweisen. Gerade die klassischen Machine Learning Verfahren (insbesondere SVM und RF) konnten gute und im Vergleich zu den Deep-Learning Verfahren bessere Resultate liefern, was sich auch mit den Ergebnissen in Habernal und Gurevych (2016) und Liebeck (2018) deckt. Die Deep-Learning Verfahren konnten hingegen nur in den sehr anspruchsvollen Sequence Tagging Experimenten bessere Resultate liefern als die klassischen Verfahren. Bezüglich der verwendeten Features konnten sich Word- und Characterembeddings, insbesondere solche mit höherer Dimensionalität (≥ 200), gegenüber den anderen Features durchsetzen (siehe insbesondere Kapitel 5.1.1). Eine Empfehlung konnte im Speziellen für die Verwendung von 200-dimensionalen Word2Vec Wordembeddings und 300-dimensionalen auf dem CommonCrawl Korpus trainierten fastText Character Embeddings ausgesprochen werden. Featurekombinationen von einzeln schlechter abschneidenden Features konnten jedoch ebenfalls gute Ergebnisse erzielen, weshalb insgesamt eine Empfehlung der Verwendung von Feature Kombinationen mit Word- und Character Embeddings ausgesprochen werden kann. Analog dazu konnten die Deep Learning Verfahren besonders dann gut abschneiden, wenn sie einen leeren, dass heißt nicht vortrainierten, Embedding Layer nutzten (siehe Kapitel 5.1.2). Dies motiviert in Zukunft eine Untersuchung von Deep Learning Verfahren mit leerem Embedding Layer oder domänenspeizifisch trainierten Embeddings.

Ebenfalls spiegelten sich die Unterschiede der Deep Learning Architekturen in den Ergebnissen wieder: Die besten Ergebnisse lieferten LSTMs mit Embedding Layer sowie CNNs mit Embedding Layer, wobei LSTMs aufgrund ihrer Berücksichtigung des Kontexts besonders für feinergranulare Klassifikation von inhaltlich gemischten Korpora zu empfehlen ist, während E-CNNs bessere Resultate auf gröberer Granularitätsebene und uniformen Datensätzen liefert. Auf diese Weise können ebenfalls Empfehlungen der Architekturen für die Verwendung in passenden Subtasks ausgesprochen werden (siehe auch Kapitel 5.2.3). Als weitere Besonderheit der Deep Learning Verfahren konnte der Einfluss der Vorverarbeitung der Daten festgestellt werden. Wurden nur Sonderzeichen (Emoticons, Bilder etc.) aus den Daten gefiltert, aber Anhäufungen von Sonderzeichen wie Ausrufezeichen beibehalten, neigten diese Klassifikatoren dazu, diese als dominante Features zu verwenden, weshalb eine Filterung der Daten für diese Klassifikatoren empfohlen wird (Kapitel 5.1.2).

Hinsichtlich der Subtasks konnte die Erkennung von argumentativen Textteilen (Subtask

A) im Mittel die besten Resultate liefern, während Subtask B (Unterscheidung von Vorschlägen, Argumenten und Positionierungen) aufgrund der größeren Anzahl an Klassen und Klassenungleichgewicht insbesondere auf Datensatz BRK deutlich schlechter ausfiel. Die restlichen Subtasks erzielten in der Regel Ergebnisse, die zwischen diesen beiden Werten lagen. Dabei konnten vor allem durch die Cross-Platform Untersuchungen interessante Zusammenhänge zwischen den statistisch sichtbaren Klassenverteilungen, den während der Codierung subjektiv wahrgenommenen Stimmungen der Diskussionen und den Klassifikationsergebnissen festgestellt werden (siehe Kapitel 5.1.4). So neigten beispielsweise Klassifikatoren dazu, auf dem subjektiv bereits als emotional negativ wahrgenommenen Korpus BRK, positive Positionierungen als negative Positionierungen zu klassifizieren. Als Lösung der Klassenverteilungen wurden drei verschiedene Oversampling-Methoden verglichen (Kapitel 5.1.1), jedoch keine signifikanten Verbesserungen der Ergebnisse erzielt.

Ein weiteres Resultat der Cross-Platform Unterschungen stellte die Unterscheidung eines Korpus bezüglich seiner Fähigkeit als Trainingsgrundlage zu dienen und der Schwierigkeit seiner Klassifikation dar. Aus diesem Grund wurde für die zukünftige Verwendung von Trainingskorpora ein standardisierter Vergleich mit einem (oder mehreren) Vergleichskorpus zur Bestimmung dieser Eigenschaften vorgeschlagen (Kapitel 5.1.3).

Für die feingranularen Untersuchungen auf Sequence Ebene wurde eine Methode zur Erzeugung von künstlichen Sätzen, welche für jedes Token zusätzlichen Kontext umfassen, vorgestellt und deren Hyperparameter getestet (Kapitel 4.5.3 und 5.2.1). Eine Verwendung von bis zu sechs zusätzlichen Token vor und nach dem eigentlich zu klassifizierenden Token konnte dabei die Leistung der Verfahren verbessern. Eine Gewichtung des originalen Tokens durch Vervielfachung konnte hingegen keinen signifikanten Nutzen erzielen. Zusätzlich wurde der Einfluss der Annotationsschemata (BIO bzw. BILOU) auf die Leistung des Sequence Taggings untersucht (Kapitel 5.2.1), wobei das BIO-Schema bessere Resultate erzielte. Ferner konnte dieses Schema selbst nach Zusammenfassung auf die originalen Labels bessere Ergebnisse liefern, als eine Klassifikation, die nur mit den originalen Labels durchgeführt wurde, was die späteren Intergranularitätsuntersuchungen weiter motivierte. Die Ergebnisse des Sequence Taggings (Kapitel 5.2) fielen insgesamt jedoch deutlich schlechter aus als jene des Sentence oder Document Taggings und lieferten vor allem eine Empfehlung für Deep Learning Klassifikatoren für diese Granularitätsebene. Zusätzlich wurden der Einfluss von Satzgrenzen in der Erstellung des Kontexts diskutiert und weitere Untersuchungen in diesem Bereich empfohlen.

Die Ergebnisse des grobgranularen Document Taggings in Kapitel 5.3 fielen ebenfalls schlechter aus als diejenigen des Sentence Taggings und motivierten vor allem die Intergranularitätsuntersuchungen. Die Zusammenfassung von Sätzen mit verschiedenen Labeln zu einem Dokument mit einem Label wurde hierbei zu einem Problem, da so teilweise irrelevanter Text mit dem jeweiligen Label versehen wird. Daher wurden hierfür Lösungen durch Duplikation oder Granularitätswechsel vorgeschlagen (Kapitel 5.3.1). Außerdem konnten Tendenzen zu einer schlechteren Klassifikation auf Testdatensatz THF erkannt werden und die Vermutung aufgestellt, dass dies in Relation zu den durchschnittlich kürzeren Beitrags- und Satzlängen dieses Korpus steht. Aus diesem Grund wurden in Kapitel 5.3.3 weitere Untersuchungen mit automatisierter Textzusammenfassung vorgeschlagen. Durch Analyse der Ergebnisse konnte eine Empfehlung des Ran-

dom Forest Klassifikators gegenüber den sonst im Mittel besser abschneidenden SVM Verfahren für Korpora mit vergleichsweise langen Sätzen (durchschnittliche Satzlänge von 155.5 für Datensatz BRK) sowie Klassifikationsaufgaben mit wenigen Klassen und einem Klassenungleichgewicht ausgesprochen werden. Zusätzlich konnte eine besondere Eignung des Deep Learning Klassifikators BLSTMs für die Verwendung auf größeren Korpora mit mehr Trainingsinstanzen festgestellt werden.

Die Ergebnisse der intergranularen Untersuchungen in Kapitel 5.4 lieferten schlussendlich nur in speziellen Fällen Verbesserungen gegenüber Klassifikationen auf einer Granularitätsebenen. Diese entsprechen dabei der intuitiven Annahme, dass die Erhöhung der Trainingsinstanzen durch die Verwendung einer feineren Granularität nur in Randfällen sinnvoll ist, in denen bereits sehr wenige Trainingsinstanzen zur Verfügung stehen. In diesem Sinne werden Intergranularitätsverfahren als Alternative oder Ergänzung zu gängigen Oversampling-Verfahren empfohlen.

Insgesamt liefert die Arbeit Empfehlungen und Abschätzung über die Leistung verschiedener Klassifikatoren und Features auf drei Granularitätsebenen. Es konnten Aussagen über die verwendeten Klassifikatoren, Features, Datensätze und Subtasks als auch über Annotationschemata, Granularitäten und Oversampling-Verfahren getroffen werden. Darüber hinaus wurden zukünftige Untersuchungen motiviert.

6.2 Ausblick

Die Evaluation der Ergebnisse konnte einige Untersuchungen, die den Umfang dieser Arbeit übersteigen, motivieren. So wurden einerseits Subtasks B, C und E auf ungefilterten Korpora durchgeführt, was durch die dadurch entstandende Mehrheitsklasse „O“ zu schlechteren Resultaten führte als die restlichen, gefilterten Untersuchungen. Eine Untersuchung von Sequence Tagging Verfahren mit Filterung und insbesondere auch die Anwendbarkeit des intergranularen Post-Processings auf den daraus gewonnen Klassifikationen ist daher anzuraten. Da in dieser Arbeit keine Untersuchung des intergranularen Trainings auf Deep Learning Klassifikatoren durchgeführt wurden und diese in den Sequence-Tagging Experimenten gute Ergebnisse lieferten, bietet sich eine zukünftige Untersuchung der Kombination von Deep Learning Klassifikatoren und intergranularem Training an.

Die Erkenntnisse bezüglich Embeddings motivieren ferner die Verwendung von leeren untrainierten Embeddinglayern in Deep Learning Untersuchungen sowie den Vergleich von domänenspezifischen Embeddings mit allgemeineren Embeddings, wie den hier in der Arbeit gut abschneidenden 300-dimensionalen word2vec Word Embeddings und 300-dimensionalen Common-Crawl fastText Character Embeddings.

Die Einführung eines standardisierten Vergleichskorpus für die Einschätzung der Qualität eines neuen Korpus bezüglich seiner Funktion als Trainingsgrundlage zu dienen könnte ebenfalls ein Teil von zukünftigen Untersuchungen darstellen. Dies ist insbesondere aufgrund der guten Ergebnisse des kombinierten Korpus für Subtasks, welche entgegengesetzte Klassenverteilungen aufweisen, für die Erstellung zukünftiger Trainingskorpora interessant.

Für zukünftige Untersuchungen im Bereich des Document Taggings könnten Verfahren

zur automatisierten Textzusammenfassung als Vorverarbeitung interessant sein, während Sequence Tagging Verfahren von einer Untersuchungen der Art von Kontexterzeugung profitieren kann. Da aufgrund der Ineffizienz der verwendeten Implementierung des CRF-Klassifikators keine Sequence-Tagging Untersuchungen für diesen durchgeführt werden konnten, wird eine weitere Untersuchung dieses Klassifikators vor allem vor dem Hintergrund der guten Resultate in Goudas et al. (2014) empfohlen.

Schließlich konnten auch inhaltliche Empfehlungen für zukünftige Beteiligungsplattformen ausgesprochen werden. So lassen die Ergebnisse den Schluss zu, dass nicht nur organisierte, sachlichere Diskussionen, sondern auch ein hinsichtlich der Klassenverteilung ausgeglichenerer Korpus entsteht, wenn die Möglichkeit zur Erstellung eigener Vorschläge gegeben ist. Zukünftige Verfahren könnten eigene Vorschläge analog zu den auf der THF-Beteiligungsplattform verwendeten Kategorien präsentieren, sodass eigene Gegenvorschläge oder Änderungen am Originalvorschlag in strukturierter Form erstellt werden und auch negative Diskussionsstränge (unsachlich, aggressiv) auf einzelne Vorschläge eingegrenzt werden.

A Anhang

In diesem Kapitel werden alle detaillierten Ergebnistabellen der klassischen Machine Learning Verfahren aufgeführt, welche die Übersichtlichkeit des Kapitels 5 einschränken würden. Dabei wurde die Struktur des Kapitels 5 grundsätzlich beibehalten, sodass eine Aufteilung in Sentence Tagging, Sequence Tagging sowie Document Tagging gegeben ist. Jede Tabelle beinhaltet die Ergebnisse eines spezifischen Datensatzes und Klassifikators.

A.1 Sentence Tagging

In diesem Kapitel folgen die Ergebnisse des Sentence Taggings.

A.1.1 Klassisches Machine Learning

Es folgen die detaillierten Sentence Tagging Ergebnisse für die Klassifikatoren Random Forest (RF), Support Vector Machine (SVM) und Support Vector Machine mit linearem Kernel (SVM-Linear) sowie Conditional Random Fields (CRF).

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	65.7	78.8	48.3	80.7	**62.5**	65.1	57.2	85.9	66.9	**81.3**
Grammatical ②	63.4	78.1	42.3	72.0	59.6	65.6	51.1	78.2	64.1	75.8
LDA Wikipedia ③	55.9	67.2	32.0	64.3	43.3	55.2	53.2	87.2	54.3	67.3
LDA Dataset ④	59.4	73.7	34.4	68.5	56.0	62.5	51.9	86.6	59.8	71.5
Character n-grams	62.0	**78.8**	35.5	**84.8**	59.8	**71.4**	54.1	**91.8**	53.1	75.6
Word Embeddings ⑤	66.1	76.2	46.0	76.9	50.2	65.1	56.6	83.1	66.7	81.1
Character Embeddings ⑥	65.5	77.6	45.8	80.6	55.4	65.6	58.1	85.9	62.9	79.0
①+②	63.8	75.5	45.3	74.6	56.6	60.9	56.2	87.3	66.0	77.9
①+③	64.4	76.7	45.5	75.6	45.1	62.5	56.5	84.3	60.5	79.4
①+④	63.8	77.6	46.2	82.2	61.1	63.5	56.5	91.2	63.1	75.6
①+⑤	**67.3**	78.0	48.4	79.1	57.3	68.2	57.5	82.4	**68.5**	81.1
①+⑥	66.1	77.4	47.6	82.2	57.1	68.8	58.4	87.7	57.5	78.8
①+②+⑤	66.6	76.7	48.1	81.1	60.9	68.8	57.8	83.7	65.1	78.1
①+②+⑥	66.0	77.6	**48.8**	81.7	51.7	66.1	58.1	87.0	55.1	76.9
①+②+⑤+⑥	67.1	78.1	45.1	76.9	52.1	66.7	**58.5**	86.8	64.9	79.6

Tabelle 46: Sentence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	77.0	77.7	**62.9**	66.4	67.5	**78.0**	**65.2**	92.5	66.4	69.0
Grammatical ②	75.8	76.2	58.0	66.2	61.8	70.2	59.5	91.9	59.2	59.9
LDA Wikipedia ③	71.8	72.2	43.1	54.2	58.6	68.0	53.2	88.4	54.4	56.3
LDA Dataset ④	75.4	76.0	47.8	57.1	62.4	67.8	54.4	86.5	52.4	52.7
Character n-grams	76.8	77.6	53.1	61.5	59.0	66.8	61.1	93.0	55.6	55.7
Word Embeddings ⑤	76.5	77.0	57.0	61.6	63.6	72.6	58.6	81.2	65.2	65.6
Character Embeddings ⑥	76.7	77.3	58.9	63.2	66.1	73.1	59.0	82.7	62.8	62.9
①+②	74.0	**78.6**	61.7	**67.6**	63.5	73.1	58.2	92.2	61.5	63.7
①+③	76.2	76.8	59.4	65.4	68.9	76.5	65.0	92.7	62.2	63.3
①+④	77.0	77.5	58.6	64.3	69.1	76.3	62.2	**93.0**	62.1	62.7
①+⑤	77.0	77.5	60.3	66.3	64.9	73.6	62.3	87.0	65.6	66.0
①+⑥	76.2	76.6	57.1	62.4	67.2	68.9	60.2	88.6	69.1	71.9
①+②+⑤	**77.1**	77.7	59.7	64.6	62.3	71.7	61.0	88.2	65.4	66.5
①+②+⑥	76.5	77.1	59.3	65.1	**70.8**	72.6	57.9	89.6	67.7	70.2
①+②+⑤+⑥	77.1	77.6	61.2	65.4	69.0	70.4	59.5	81.8	**72.8**	**74.3**

Tabelle 47: Sentence-Tagging Ergebnisse für Datensatz **Both** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	76.6	77.4	**62.7**	67.4	63.5	72.9	**64.2**	93.0	60.0	61.4
Grammatical ②	76.2	76.9	57.9	65.8	59.5	70.5	58.5	90.9	57.9	60.8
LDA Wikipedia ③	71.9	72.2	43.8	53.2	57.6	69.5	54.0	90.8	52.1	57.8
LDA Dataset ④	76.0	77.1	46.9	53.2	56.1	67.6	54.2	85.4	57.4	59.1
Character n-grams	76.1	76.9	56.3	62.9	57.4	69.7	57.0	**93.6**	58.7	61.2
Word Embeddings ⑤	77.1	77.9	59.9	64.5	59.5	71.2	60.4	84.5	65.1	68.4
Character Embeddings ⑥	77.0	77.6	60.5	65.4	56.6	69.5	58.8	83.3	67.2	70.7
①+②	76.9	77.6	62.3	**68.6**	64.8	**74.3**	61.4	92.6	55.2	60.3
①+③	76.1	76.7	61.6	66.3	**65.1**	73.6	63.6	92.7	61.8	65.2
①+④	77.0	77.8	62.0	66.3	63.8	74.3	61.9	92.6	61.3	64.1
①+⑤	77.1	77.9	61.9	68.1	54.7	72.6	63.1	90.6	64.7	67.7
①+⑥	76.9	77.6	60.3	65.1	60.0	71.4	58.6	83.5	66.4	69.6
①+②+⑤	**77.3**	**78.1**	61.0	66.1	57.4	73.4	61.6	91.5	66.5	69.6
①+②+⑥	77.0	77.7	62.3	68.5	56.9	71.2	59.5	83.6	**68.7**	**72.8**
①+②+⑤+⑥	77.0	77.9	62.4	67.3	58.7	72.2	62.3	85.7	65.1	68.8

Tabelle 48: Sentence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	51.2	80.2	38.4	85.6	67.1	69.3	59.7	90.3	63.3	80.3
Grammatical ②	62.1	70.9	42.5	67.9	56.3	59.9	52.6	87.8	61.8	72.0
LDA Wikipedia ③	56.9	68.1	34.8	62.0	54.8	58.3	53.8	85.6	55.3	64.8
LDA Dataset ④	56.6	67.2	34.6	68.4	61.6	65.6	50.6	80.0	56.2	69.4
Character n-grams	61.4	75.8	36.2	54.8	57.3	66.7	50.4	69.7	59.7	69.4
Word Embeddings ⑤	65.7	74.0	50.7	77.7	68.1	71.9	58.3	85.5	**72.1**	**81.5**
Character Embeddings ⑥	**66.8**	75.6	50.7	78.7	58.1	67.7	58.5	81.2	71.2	81.3
①+②	60.4	**81.6**	39.9	**85.8**	67.9	70.3	59.2	89.9	61.2	77.5
①+③	58.9	80.6	50.5	83.5	65.8	68.2	58.9	90.1	63.2	77.7
①+④	59.2	80.7	48.9	81.7	68.5	70.8	**60.0**	90.2	64.0	78.3
①+⑤	49.7	80.1	49.2	84.4	67.0	69.8	56.9	90.8	65.5	79.2
①+⑥	64.0	74.6	**51.4**	82.2	61.4	61.5	57.8	84.5	61.4	76.4
①+②+⑤	64.8	80.0	48.9	82.9	**69.5**	**72.4**	56.8	**90.9**	56.7	79.0
①+②+⑥	64.7	72.7	51.2	82.3	62.2	62.5	56.8	83.2	43.0	75.6
①+②+⑤+⑥	64.9	75.6	51.3	82.3	39.2	64.6	57.4	90.0	62.7	78.3

Tabelle 49: Sentence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	76.3	76.6	58.9	66.5	68.1	75.5	67.3	92.4	64.4	64.8
Grammatical ②	74.9	75.4	56.9	63.3	62.2	68.8	56.5	88.1	61.9	63.1
LDA Wikipedia ③	71.9	72.2	41.6	51.9	57.9	64.2	54.0	90.7	54.0	54.4
LDA Dataset ④	75.9	77.1	47.4	56.7	59.6	64.4	54.4	85.4	56.7	58.9
Character n-grams	76.3	77.1	53.7	59.6	51.9	72.2	54.1	75.4	62.2	65.4
Word Embeddings ⑤	76.9	77.4	58.5	65.7	69.4	74.1	60.9	88.5	**75.0**	**76.8**
Character Embeddings ⑥	**77.5**	**77.9**	58.0	62.0	73.5	76.3	61.7	89.7	69.9	71.5
①+②	76.5	76.8	60.8	67.2	66.8	75.1	67.6	91.6	71.3	72.6
①+③	76.4	76.7	62.7	66.5	42.8	72.4	67.1	92.5	64.2	64.6
①+④	76.4	76.7	62.7	66.7	68.5	75.8	67.2	92.4	70.0	71.5
①+⑤	76.0	77.2	61.3	67.2	73.7	**79.9**	60.8	**94.0**	70.7	72.2
①+⑥	76.2	76.8	61.3	66.0	**76.2**	77.9	65.3	92.3	66.4	66.6
①+②+⑤	76.5	76.8	65.0	69.0	68.1	76.3	68.0	92.2	68.5	69.8
①+②+⑥	76.2	76.8	61.8	66.7	74.0	75.7	65.7	92.4	72.7	73.1
①+②+⑤+⑥	76.5	76.8	**65.6**	**69.4**	73.7	75.2	**68.1**	92.3	73.4	74.0

Tabelle 50: Sentence-Tagging Ergebnisse für Datensatz **Both** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	76.3	76.8	43.9	61.9	42.8	72.4	66.9	92.0	40.8	65.6
Grammatical ②	76.0	76.9	57.6	63.9	61.0	67.6	57.2	87.9	59.9	62.0
LDA Wikipedia ③	71.9	72.3	43.4	52.6	57.9	69.5	53.5	83.8	55.0	61.0
LDA Dataset ④	76.1	77.2	45.9	55.3	57.7	62.2	52.7	83.4	57.8	59.7
Character n-grams	76.0	77.0	49.6	54.2	42.8	72.4	57.9	83.2	60.0	62.9
Word Embeddings ⑤	77.0	77.7	61.3	66.2	68.4	73.1	60.1	89.6	**72.1**	**73.2**
Character Embeddings ⑥	**77.2**	**77.8**	62.7	67.3	66.9	74.1	60.2	84.8	69.8	71.5
①+②	76.8	77.4	62.0	68.2	65.1	75.5	66.9	92.1	65.7	70.0
①+③	76.0	77.2	61.3	66.0	42.8	72.4	66.6	92.0	40.1	65.4
①+④	76.2	76.7	61.8	67.4	65.8	76.0	66.6	91.9	68.0	71.5
①+⑤	76.0	77.3	62.7	68.2	**70.9**	**78.2**	67.2	92.1	68.2	72.2
①+⑥	76.6	77.4	63.8	67.7	61.1	65.9	66.5	92.1	65.4	68.4
①+②+⑤	77.0	77.6	62.8	68.4	67.9	77.0	**67.3**	92.2	64.6	68.8
①+②+⑥	76.6	77.4	64.8	69.0	61.7	66.6	66.8	92.1	65.8	68.6
①+②+⑤+⑥	76.6	77.4	**65.6**	**69.7**	68.3	77.5	67.1	**92.2**	65.8	68.6

Tabelle 51: Sentence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	62.0	75.8	49.3	82.9	67.1	69.3	60.3	89.1	62.7	80.0
Grammatical ②	62.5	71.4	38.5	56.7	57.6	63.5	48.2	70.2	62.6	74.7
LDA Wikipedia ③	54.8	73.1	34.8	62.8	49.4	50.5	54.5	82.7	57.1	66.9
LDA Dataset ④	53.4	74.4	35.8	58.4	59.8	64.6	42.4	58.0	48.5	57.1
Character n-grams	58.1	64.2	33.0	47.6	60.2	63.5	49.2	66.9	56.0	64.1
Word Embeddings ⑤	65.3	74.8	46.0	69.9	60.9	66.7	52.7	71.9	72.1	81.1
Character Embeddings ⑥	**66.5**	75.9	47.7	71.0	66.4	70.8	55.2	77.7	**72.3**	**81.3**
①+②	62.9	**76.0**	51.5	79.3	68.3	70.8	60.1	88.7	64.1	79.8
①+③	62.7	75.6	50.7	79.1	68.2	70.3	60.2	89.0	62.7	78.3
①+④	61.9	74.9	50.6	78.8	69.0	71.4	**60.4**	89.0	62.5	78.1
①+⑤	65.1	74.0	50.3	83.3	66.8	69.8	59.1	89.2	63.8	79.2
①+⑥	61.8	75.7	52.0	80.7	68.1	71.4	56.8	90.9	50.1	50.7
①+②+⑤	62.3	75.8	49.9	**83.4**	**69.5**	**72.4**	57.4	**91.0**	61.6	77.9
①+②+⑥	64.9	74.1	52.4	80.8	42.8	45.8	56.8	90.9	43.0	75.6
①+②+⑤+⑥	65.6	74.9	**52.7**	80.9	68.1	71.4	57.1	91.0	64.1	78.8

Tabelle 52: Sentence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	72.8	72.8	60.2	65.3	72.8	**78.9**	66.2	**90.8**	70.9	72.4
Grammatical ②	75.3	76.1	53.4	58.7	62.0	68.0	50.8	68.8	61.3	63.1
LDA Wikipedia ③	68.9	68.9	37.2	41.9	57.3	64.4	50.8	74.1	57.8	62.4
LDA Dataset ④	60.1	60.4	37.9	42.4	61.7	65.4	47.9	65.8	58.8	60.5
Character n-grams	71.8	71.8	43.7	47.3	61.4	67.3	54.4	75.7	62.8	65.8
Word Embeddings ⑤	77.0	77.6	54.9	58.8	68.7	73.8	59.0	83.2	72.0	74.3
Character Embeddings ⑥	**77.6**	**78.1**	57.2	60.7	72.5	77.5	**75.8**	76.4	72.7	74.3
①+②	75.6	75.8	62.9	66.8	71.6	77.5	66.3	90.6	72.0	73.2
①+③	73.7	73.7	60.2	63.6	72.0	78.2	66.6	90.1	**73.2**	**74.7**
①+④	73.1	73.2	60.1	65.3	72.6	78.9	66.4	90.7	71.1	72.8
①+⑤	75.9	76.1	62.9	66.4	72.0	78.2	66.8	90.8	71.5	73.0
①+⑥	74.9	75.1	60.5	71.7	66.7	71.4	65.2	89.6	70.7	71.2
①+②+⑤	75.6	75.9	**63.9**	67.6	71.4	77.5	66.9	90.6	67.8	69.0
①+②+⑥	74.9	75.1	61.0	**72.6**	**74.7**	76.4	64.9	89.6	71.2	71.4
①+②+⑤+⑥	75.5	75.7	63.9	67.4	72.0	77.7	66.6	90.6	71.0	71.9

Tabelle 53: Sentence-Tagging Ergebnisse für Datensatz **Both** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	72.7	72.7	60.0	64.8	67.9	76.3	66.1	90.7	67.6	70.9
Grammatical ②	76.2	77.2	53.5	59.1	59.7	66.3	50.0	67.1	54.8	54.9
LDA Wikipedia ③	68.2	68.2	38.4	43.7	56.0	70.5	54.3	88.5	54.8	61.2
LDA Dataset ④	60.2	60.7	39.1	43.7	55.9	60.5	51.8	76.0	58.9	61.2
Character n-grams	74.2	74.3	46.8	50.7	60.6	66.8	53.3	73.3	63.1	65.2
Word Embeddings ⑤	76.5	77.5	60.7	65.6	64.8	70.5	55.7	76.4	**68.7**	69.8
Character Embeddings ⑥	**77.2**	**78.0**	61.8	65.9	67.8	74.6	57.6	81.5	66.6	68.6
①+②	73.2	73.4	62.5	67.0	65.6	75.3	66.1	90.6	65.6	69.8
①+③	72.6	72.6	59.8	64.7	67.2	76.3	66.5	91.0	65.9	69.2
①+④	72.8	72.8	60.5	65.0	67.7	76.5	66.1	90.7	68.4	**72.2**
①+⑤	76.4	76.8	61.9	66.4	**69.9**	**77.7**	66.7	91.0	67.7	71.3
①+⑥	76.4	77.4	62.5	66.6	69.2	77.2	64.3	**92.4**	63.9	67.3
①+②+⑤	76.7	77.2	63.3	67.7	67.4	76.5	**67.1**	91.0	67.2	70.9
①+②+⑥	77.1	77.6	62.3	66.9	68.8	77.2	66.5	90.9	67.5	70.5
①+②+⑤+⑥	74.0	74.2	**64.8**	**68.8**	68.3	77.5	66.9	91.1	64.9	68.4

Tabelle 54: Sentence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	59.9	76.6	41.3	81.2	51.8	58.3	31.5	46.1	54.1	70.8
Grammatical ②	61.3	77.0	41.5	79.4	55.2	60.9	51.7	88.5	52.7	68.0
LDA Wikipedia ③	60.6	76.9	41.4	81.0	54.0	59.7	50.7	89.3	57.9	72.6
LDA Dataset ④	60.8	77.2	41.4	81.0	56.5	62.0	50.7	89.0	57.9	72.6
Character n-grams	60.6	76.9	41.4	81.0	56.5	62.0	50.7	89.0	**58.7**	**72.9**
Word Embeddings ⑤	61.8	77.8	**44.0**	**81.9**	53.9	59.7	**53.7**	**89.5**	56.8	72.5
Character Embeddings ⑥	**62.9**	**78.0**	30.6	78.2	54.5	61.1	50.2	89.3	49.0	68.3
①+②	60.5	77.2	39.2	79.3	55.4	59.6	52.3	87.9	52.8	67.6
①+③	59.0	77.1	41.3	81.2	55.3	61.4	31.5	46.1	52.3	69.8
①+④	59.2	77.0	41.3	81.2	56.9	**62.4**	30.0	43.4	50.4	69.0
①+⑤	56.7	76.4	42.3	81.6	53.9	59.7	44.1	67.4	52.8	70.4
①+⑥	61.4	77.7	30.6	78.0	46.7	56.1	50.3	77.8	45.9	66.2
①+②+⑤	60.0	76.3	42.6	80.1	54.7	60.6	51.3	88.5	52.1	66.8
①+②+⑥	62.0	77.3	31.0	78.1	54.1	60.2	50.6	89.2	52.4	68.2
①+②+⑤+⑥	62.8	77.6	40.1	79.8	**57.0**	62.3	50.7	89.2	55.6	70.2

Tabelle 55: Sentence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **CRF**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	70.3	73.5	54.7	71.4	61.8	64.5	19.9	25.7	61.5	62.5
Grammatical ②	70.4	74.8	55.4	72.0	58.5	60.5	55.0	89.8	62.7	63.4
LDA Wikipedia ③	71.1	75.6	55.9	72.1	62.1	64.3	54.7	90.4	59.8	61.0
LDA Dataset ④	71.1	75.6	55.8	72.1	62.1	64.3	54.7	90.4	59.8	61.0
Character n-grams	71.1	75.6	55.8	72.1	**66.7**	**68.0**	54.7	90.4	63.6	64.4
Word Embeddings ⑤	71.3	75.6	56.5	72.2	64.1	66.4	55.3	90.3	62.6	63.6
Character Embeddings ⑥	**71.6**	**76.1**	47.6	71.0	60.8	63.5	54.6	90.4	62.9	63.9
①+②	71.3	75.8	55.5	72.3	60.1	61.9	55.9	89.2	61.6	62.5
①+③	70.5	74.7	54.6	71.1	61.8	64.5	20.4	26.5	60.1	61.3
①+④	71.4	76.0	54.6	71.2	61.8	64.5	22.4	30.4	60.8	61.8
①+⑤	71.3	75.8	57.4	73.3	61.9	64.9	27.8	39.3	**63.6**	**64.5**
①+⑥	71.5	76.1	46.9	70.8	60.8	63.2	53.5	80.3	58.5	59.8
①+②+⑤	71.5	75.9	54.7	70.8	61.3	63.4	56.9	89.3	58.7	59.6
①+②+⑥	69.3	74.0	47.7	70.7	59.3	61.3	55.8	90.2	62.3	63.2
①+②+⑤+⑥	69.2	74.0	**57.6**	**74.0**	60.3	62.3	**56.9**	**90.6**	62.9	63.6

Tabelle 56: Sentence-Tagging Ergebnisse für Datensatz **Both** mit Klassifikator **CRF**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	**72.6**	**73.5**	59.5	64.1	58.9	69.8	57.7	**91.8**	59.9	64.5
Grammatical ②	71.7	72.4	60.8	65.4	53.2	65.6	57.5	90.9	53.8	58.0
LDA Wikipedia ③	72.3	73.1	59.7	64.3	**62.4**	**71.9**	58.2	91.7	59.0	63.3
LDA Dataset ④	72.3	73.1	59.1	63.8	62.4	71.9	58.2	91.7	60.7	**64.8**
Character n-grams	72.3	73.1	59.7	64.3	58.3	68.4	58.4	91.7	59.0	63.3
Word Embeddings ⑤	71.9	72.7	**61.6**	**66.9**	55.8	67.7	**58.7**	91.8	**61.2**	64.8
Character Embeddings ⑥	70.5	71.1	53.4	64.7	55.7	67.2	50.1	90.6	59.8	64.8
①+②	71.6	72.3	60.3	65.9	56.9	66.9	57.4	91.4	55.1	58.8
①+③	71.1	71.8	59.4	64.0	57.8	68.1	57.7	91.8	57.8	62.2
①+④	71.1	71.8	59.4	63.9	58.9	69.8	56.3	91.6	59.9	64.5
①+⑤	71.9	72.7	61.2	66.1	55.8	67.7	57.9	91.6	58.5	63.0
①+⑥	70.6	71.3	47.2	61.7	54.1	66.4	57.0	91.5	59.1	64.3
①+②+⑤	71.3	72.0	58.5	63.4	58.1	67.7	56.4	90.7	53.8	58.3
①+②+⑥	71.2	71.9	57.4	65.9	59.2	68.6	58.2	91.0	55.3	59.3
①+②+⑤+⑥	71.4	72.1	57.4	65.8	60.4	69.4	56.1	91.0	55.2	59.2

Tabelle 57: Sentence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **CRF**.

A.1.2 Klassisches Machine Learning: Cross-Platform

In diesem Kapitel werden die Cross-Platform Ergebnisse betrachtet. In allen Tabellen sowie Abbildungen wird jeweils der Datensatz des verwendeten Trainingssets angegeben. Der Datensatz des verwendeten Testsets ist für Cross-Platform Untersuchungen immer der jeweils andere Datensatz. Der kombinierte Datensatz Both wurde hierbei nicht untersucht.

Es folgen die detaillierten Cross-Platform Ergebnisse für die Klassifikatoren Random Forest (RF), Support Vector Machine (SVM) und Support Vector Machine mit linearem Kernel (SVM-Linear) sowie Conditional Random Fields (CRF).

Subtask	A		B		C		D		E	
*F*1-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	72.8	72.8	32.2	52.2	**60.8**	**66.6**	**60.7**	87.7	45.5	46.8
Grammatical ②	72.9	73.1	40.4	53.7	58.6	63.0	55.1	81.6	53.8	53.8
LDA Wikipedia ③	68.6	68.6	29.9	42.5	49.3	49.4	53.9	89.8	49.4	49.4
LDA Dataset ④	64.9	64.9	29.2	36.1	42.1	42.1	36.4	50.0	52.8	53.6
Character n-grams	**74.6**	**74.9**	29.1	53.5	53.5	55.2	40.9	61.3	42.4	44.1
Word Embeddings ⑤	68.4	68.4	37.4	52.9	48.2	48.2	39.9	53.9	53.9	54.2
Character Embeddings ⑥	71.4	71.4	37.1	54.5	56.2	57.6	40.2	56.4	48.2	48.9
①+②	70.4	70.4	38.0	51.2	58.1	62.0	59.3	89.2	**55.9**	**55.9**
①+③	71.8	71.8	36.6	50.9	47.9	47.9	59.1	87.9	36.7	40.9
①+④	70.7	70.7	38.7	52.8	50.3	55.2	58.9	**91.3**	54.6	54.9
①+⑤	73.2	73.4	36.3	52.0	56.6	62.7	59.0	84.6	55.2	55.3
①+⑥	71.2	71.3	39.8	50.0	51.8	52.1	59.4	88.8	49.8	50.4
①+②+⑤	69.6	69.6	41.6	53.5	53.4	56.4	59.2	85.9	55.9	55.9
①+②+⑥	71.3	71.4	40.5	**55.6**	59.8	64.4	40.7	57.3	55.5	55.5
①+②+⑤+⑥	71.7	71.8	**42.0**	54.8	50.9	51.8	40.7	56.7	53.6	53.8

Tabelle 58: Cross-Platform Sentence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	57.8	**81.2**	39.8	73.9	64.5	64.6	56.6	90.8	49.9	58.0
Grammatical ②	54.1	79.2	37.1	69.6	60.9	60.9	52.5	89.0	57.8	62.0
LDA Wikipedia ③	54.8	73.3	31.3	55.6	40.0	41.1	50.5	86.7	49.4	52.2
LDA Dataset ④	55.6	75.0	30.6	51.8	38.5	39.6	48.1	76.4	54.5	58.2
Character n-grams	56.5	80.6	35.9	77.1	40.9	41.1	51.4	**92.6**	56.1	61.1
Word Embeddings ⑤	57.6	80.6	38.1	69.2	41.2	43.8	56.1	81.3	66.2	73.0
Character Embeddings ⑥	58.5	80.4	37.6	68.4	38.9	40.6	57.6	82.0	63.4	69.6
①+②	56.2	79.5	**40.8**	76.7	61.8	62.0	54.5	90.9	55.6	59.2
①+③	57.8	79.6	38.7	69.1	**65.1**	**65.1**	57.3	90.0	52.1	57.1
①+④	58.9	75.3	38.0	75.1	56.2	56.8	55.0	89.6	52.8	56.3
①+⑤	57.9	80.6	37.8	68.5	47.8	48.4	57.0	89.9	66.5	71.5
①+⑥	57.9	80.2	37.1	67.8	58.0	58.3	57.4	81.6	62.7	68.8
①+②+⑤	57.9	80.8	38.9	70.2	46.2	47.9	55.9	80.3	**68.4**	**74.5**
①+②+⑥	57.6	80.4	40.1	**77.8**	43.2	45.3	57.9	82.2	59.4	62.4
①+②+⑤+⑥	**59.6**	80.5	39.3	72.6	43.2	45.3	**59.1**	83.2	63.4	69.4

Tabelle 59: Cross-Platform Sentence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	75.8	77.0	25.3	53.5	63.4	66.8	60.4	91.4	43.7	46.0
Grammatical ②	68.2	68.3	**41.2**	**54.6**	57.4	62.0	38.3	58.0	57.0	57.4
LDA Wikipedia ③	69.1	69.1	29.3	45.7	52.1	55.4	54.5	90.1	50.5	50.6
LDA Dataset ④	62.9	63.1	28.8	36.2	54.5	56.4	34.2	44.8	53.6	53.8
Character n-grams	70.7	70.9	36.4	41.0	54.7	57.1	33.8	41.9	59.6	60.3
Word Embeddings ⑤	65.2	65.4	35.6	53.2	58.8	59.8	39.3	55.6	61.2	61.4
Character Embeddings ⑥	69.0	69.1	39.8	49.7	55.2	56.2	39.5	54.9	**61.3**	**61.4**
①+②	44.0	58.1	25.2	53.6	62.1	65.1	**60.6**	91.3	47.1	47.9
①+③	75.5	76.3	34.6	53.6	62.9	66.3	60.2	91.4	46.5	47.7
①+④	**75.9**	**77.1**	33.5	52.7	**63.8**	66.1	60.3	91.4	46.3	47.5
①+⑤	75.7	77.0	33.9	53.3	61.8	64.4	58.7	**92.5**	46.4	47.5
①+⑥	71.6	71.7	38.2	52.2	57.6	70.0	59.8	87.1	56.1	56.1
①+②+⑤	74.5	74.6	32.5	53.2	61.8	64.2	58.2	92.4	35.0	40.3
①+②+⑥	68.4	68.4	38.5	52.2	60.1	**70.2**	59.9	86.6	56.1	56.1
①+②+⑤+⑥	71.4	71.5	38.5	52.3	59.1	62.0	60.3	92.1	56.1	56.1

Tabelle 60: Cross-Platform Sentence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	59.9	79.0	31.0	**83.9**	26.2	35.4	**58.5**	89.1	20.3	24.6
Grammatical ②	52.2	79.8	36.7	60.5	**60.3**	**62.0**	51.6	82.3	56.4	60.5
LDA Wikipedia ③	54.5	73.4	31.4	55.8	41.5	42.7	50.6	76.7	41.1	41.8
LDA Dataset ④	55.5	74.3	33.0	63.7	54.1	54.7	48.6	78.2	52.6	56.7
Character n-grams	53.6	80.7	33.9	58.2	27.1	35.9	52.5	78.3	55.2	59.0
Word Embeddings ⑤	59.7	79.6	40.0	67.5	56.3	56.8	57.5	85.9	**72.0**	**80.5**
Character Embeddings ⑥	60.4	80.3	**42.5**	70.1	50.1	50.5	54.3	75.8	63.7	69.2
①+②	58.1	80.2	40.3	79.9	44.9	47.4	57.1	89.3	56.6	59.2
①+③	50.2	80.2	40.1	69.0	26.2	35.4	57.8	89.0	20.6	25.1
①+④	60.2	79.1	40.1	79.3	44.9	47.4	58.1	88.8	53.5	56.1
①+⑤	49.8	80.3	40.6	79.9	51.3	52.6	57.6	89.5	58.4	60.9
①+⑥	**62.2**	78.3	39.8	66.0	49.4	49.5	57.6	**89.8**	57.7	63.9
①+②+⑤	59.1	76.3	40.2	80.1	51.5	52.6	57.0	89.5	57.1	59.4
①+②+⑥	56.1	**80.8**	39.9	67.3	49.4	49.5	57.0	89.6	57.9	62.8
①+②+⑤+⑥	58.8	74.4	40.6	68.2	49.4	49.5	56.7	89.7	58.2	63.3

Tabelle 61: Cross-Platform Sentence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	71.4	71.5	33.4	53.1	62.5	65.6	60.8	90.2	43.0	45.4
Grammatical ②	68.6	68.6	40.2	46.6	57.4	61.3	35.1	43.9	51.9	51.9
LDA Wikipedia ③	40.5	54.5	29.8	46.4	54.0	64.6	55.1	88.1	38.6	40.3
LDA Dataset ④	49.7	52.6	33.6	36.8	54.3	55.9	49.8	76.8	49.7	49.8
Character n-grams	61.6	62.1	35.3	38.6	50.8	55.7	51.4	73.0	57.9	59.3
Word Embeddings ⑤	66.3	66.3	**41.4**	49.6	57.5	58.4	55.1	74.8	**63.4**	**63.7**
Character Embeddings ⑥	69.7	69.7	41.4	49.4	48.9	50.8	37.9	49.5	62.1	62.2
①+②	71.4	71.5	35.4	51.4	62.1	65.1	41.7	59.1	46.3	47.7
①+③	71.2	71.3	38.1	52.9	63.4	**66.3**	60.6	90.1	44.4	46.2
①+④	71.1	71.2	39.2	53.0	**63.9**	66.1	**61.3**	90.4	44.2	46.0
①+⑤	67.8	67.9	33.8	53.2	61.8	64.2	61.1	91.3	42.4	44.7
①+⑥	71.3	71.5	38.4	53.4	60.9	62.5	58.2	**92.6**	58.1	58.2
①+②+⑤	71.7	71.9	32.6	53.3	61.8	64.2	58.4	92.3	42.3	44.5
①+②+⑥	69.5	69.5	39.0	**53.5**	53.6	55.4	58.2	92.6	57.1	57.4
①+②+⑤+⑥	**71.8**	**72.0**	38.4	53.3	60.1	62.2	58.3	92.5	57.1	57.4

Tabelle 62: Cross-Platform Sentence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	57.0	68.7	38.8	66.4	52.2	53.1	**58.2**	87.7	58.3	61.4
Grammatical ②	51.8	80.1	33.6	51.6	**56.9**	**57.3**	40.6	50.4	61.6	70.7
LDA Wikipedia ③	51.2	60.9	25.3	38.1	39.3	41.7	52.8	82.4	40.4	41.0
LDA Dataset ④	46.9	50.5	22.0	27.4	48.2	50.0	44.6	62.8	52.4	56.1
Character n-grams	60.9	77.6	32.1	52.7	50.2	50.5	44.2	57.5	59.7	64.3
Word Embeddings ⑤	55.5	**81.1**	39.6	66.6	49.2	49.5	50.7	67.2	**67.8**	**75.6**
Character Embeddings ⑥	56.4	80.6	39.0	67.0	51.7	52.1	50.4	69.9	59.8	65.2
①+②	56.2	68.4	40.8	68.1	51.5	52.6	57.1	87.2	56.7	59.2
①+③	56.8	68.4	39.1	66.8	49.6	51.0	57.9	87.5	56.8	59.7
①+④	57.2	68.7	39.5	66.1	50.9	52.1	58.0	87.1	55.2	57.7
①+⑤	**61.1**	77.3	40.0	67.7	53.2	54.2	58.2	88.0	58.0	60.7
①+⑥	60.6	73.6	39.8	67.5	52.6	53.6	56.4	**90.3**	58.6	67.7
①+②+⑤	58.9	77.1	40.8	68.5	51.5	52.6	57.5	88.1	56.9	59.2
①+②+⑥	59.6	76.1	**41.2**	68.8	52.8	53.6	56.7	87.7	59.2	68.2
①+②+⑤+⑥	57.0	70.6	40.8	**69.0**	50.9	52.1	56.9	88.3	59.2	68.2

Tabelle 63: Cross-Platform Sentence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	46.0	49.9	27.8	39.5	52.8	55.2	49.4	90.5	41.0	37.6
Grammatical ②	58.5	60.3	**33.2**	**43.7**	56.9	61.1	51.2	88.9	43.5	41.4
LDA Wikipedia ③	49.3	52.7	33.1	41.5	51.8	54.7	50.6	90.5	43.5	40.6
LDA Dataset ④	46.8	50.6	33.0	41.4	55.4	58.9	49.7	90.3	43.5	40.6
Character n-grams	49.1	52.6	33.1	41.5	55.4	58.9	49.7	90.3	44.7	42.1
Word Embeddings ⑤	46.8	50.5	31.3	41.3	54.5	57.9	51.2	90.4	39.6	35.3
Character Embeddings ⑥	47.4	51.0	23.4	36.8	51.9	54.1	50.7	90.5	31.5	25.1
①+②	50.4	53.3	29.3	41.8	**57.3**	**63.3**	**53.4**	90.4	**47.6**	**45.9**
①+③	50.0	53.3	27.8	39.4	54.9	57.8	49.4	90.5	40.0	35.9
①+④	50.0	53.3	27.8	39.5	54.6	57.9	49.6	90.5	38.5	34.2
①+⑤	56.9	58.7	27.6	39.8	54.5	57.9	50.1	90.4	37.0	31.9
①+⑥	49.6	52.8	23.3	36.8	38.3	34.9	49.6	90.3	37.5	33.0
①+②+⑤	57.2	59.0	33.0	42.7	57.2	61.3	51.1	89.0	45.4	43.5
①+②+⑥	**61.9**	**63.1**	26.9	41.2	56.0	60.2	51.7	90.4	42.8	39.6
①+②+⑤+⑥	59.2	60.8	29.1	41.2	56.0	61.6	51.3	**90.6**	44.6	41.9

Tabelle 64: Cross-Platform Sentence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **CRF**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	56.1	**75.5**	38.7	75.4	38.2	34.2	13.9	14.4	47.6	52.4
Grammatical ②	47.8	72.5	39.0	75.4	41.2	39.5	51.9	88.3	45.1	50.5
LDA Wikipedia ③	53.9	74.7	39.1	76.0	41.9	39.2	51.4	89.0	45.2	49.5
LDA Dataset ④	53.9	74.7	38.9	75.8	41.9	39.2	51.4	89.0	48.1	53.0
Character n-grams	53.9	74.7	39.1	76.0	43.1	41.0	51.4	89.0	45.2	49.5
Word Embeddings ⑤	52.4	74.2	**39.8**	76.7	40.8	37.6	52.2	**89.3**	46.8	51.5
Character Embeddings ⑥	45.3	71.5	33.6	**78.4**	39.6	36.5	48.1	89.2	45.7	48.0
①+②	46.0	71.8	38.7	73.7	43.4	43.7	52.6	88.5	45.9	51.0
①+③	**57.6**	73.2	38.8	75.4	43.7	41.7	13.9	14.4	46.2	51.5
①+④	47.7	72.5	38.8	75.4	38.2	34.2	12.9	12.5	47.6	52.4
①+⑤	54.1	74.8	39.8	76.1	40.2	36.8	23.3	31.4	49.6	54.2
①+⑥	46.3	71.9	32.1	78.3	37.4	33.6	34.1	51.1	47.0	49.8
①+②+⑤	46.1	71.8	38.4	72.2	41.4	41.4	**53.0**	86.7	45.5	50.9
①+②+⑥	46.1	71.8	35.6	78.1	43.4	43.7	49.2	78.8	45.0	51.1
①+②+⑤+⑥	45.9	71.7	35.2	77.8	**44.4**	**45.1**	50.5	80.7	**50.8**	**56.3**

Tabelle 65: Cross Platform Sentence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **CRF**.

A.2 Sequence Tagging

In diesem Kapitel werden die detaillierten Ergebnisse des Sequence Taggings aufgeführt.

A.2.1 Klassisches Machine Learning

Es folgen die detaillierten Sequence Tagging Ergebnisse für die Klassifikatoren Random Forest (RF), Support Vector Machine (SVM) und Support Vector Machine mit linearem Kernel (SVM-Linear) sowie Conditional Random Fields (CRF).

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	32.6	53.3	13.7	30.8	11.6	27.9	18.9	38.7	**23.2**	38.8
Grammatical ②	31.6	55.7	11.7	25.4	14.5	39.8	14.4	26.5	20.9	33.1
LDA Wikipedia ③	18.8	48.1	2.4	2.8	2.2	1.9	4.6	6.7	5.3	4.7
LDA Dataset ④	8.0	8.8	2.4	1.0	3.9	4.9	8.6	16.7	14.6	38.3
Character n-grams	**33.8**	54.0	**17.9**	**68.0**	19.3	**93.0**	**29.8**	**91.4**	21.8	**39.5**
Word Embeddings ⑤	21.6	34.0	10.9	17.6	**19.3**	92.5	20.5	48.6	18.9	36.8
Character Embeddings ⑥	28.6	46.6	10.7	18.1	15.1	42.2	21.1	45.9	21.8	29.9
①+②	28.1	44.7	15.5	33.5	8.6	17.2	13.9	24.9	19.7	32.6
①+③	25.2	35.6	6.8	13.5	17.9	66.2	4.7	5.2	21.9	32.7
①+④	33.0	54.3	13.5	31.1	12.0	31.5	23.0	58.9	20.3	36.2
①+⑤	31.7	**56.9**	10.1	17.8	17.7	62.4	6.2	7.8	19.9	27.6
①+⑥	25.7	41.2	11.4	21.9	15.2	51.1	20.4	44.8	21.7	31.3
①+②+⑤	24.1	35.0	12.4	26.4	18.3	57.9	12.6	21.3	19.7	28.7
①+②+⑥	24.8	37.5	12.0	24.8	19.2	63.8	15.5	28.5	19.4	27.4
①+②+⑤+⑥	25.5	38.5	12.5	22.8	16.4	47.1	14.3	25.3	19.8	29.2

Tabelle 66: Sequence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	28.6	**55.4**	20.5	52.3	23.7	91.0	19.4	44.9	21.6	34.4
Grammatical ②	**31.2**	54.1	21.0	49.0	22.2	91.2	27.6	86.8	23.1	40.8
LDA Wikipedia ③	18.7	51.5	2.9	3.0	2.1	1.5	24.4	88.8	4.8	4.9
LDA Dataset ④	10.2	14.2	4.0	2.6	3.2	2.0	7.7	11.7	13.2	24.5
Character n-grams	-	-	**22.9**	50.3	23.3	91.6	**29.4**	92.4	22.0	41.6
Word Embeddings ⑤	28.6	54.9	15.1	31.6	22.6	91.7	20.3	51.1	21.9	44.5
Character Embeddings ⑥	26.3	43.8	15.3	29.9	10.6	22.1	21.6	51.7	22.0	43.8
①+②	-	-	21.9	42.6	20.6	66.6	25.6	71.9	20.6	29.2
①+③	30.3	54.2	20.3	41.4	18.9	56.0	18.2	41.2	23.0	40.0
①+④	28.3	44.7	22.4	51.3	24.0	86.5	19.3	43.5	22.2	45.2
①+⑤	19.7	27.6	21.9	50.6	23.1	91.5	27.0	88.8	15.1	22.0
①+⑥	22.9	34.5	21.9	56.9	23.7	90.4	27.0	**92.8**	20.1	27.1
①+②+⑤	20.7	31.1	21.9	46.4	23.4	**91.7**	25.9	74.2	23.1	44.3
①+②+⑥	21.7	34.8	22.1	**58.2**	15.3	41.4	27.3	86.9	21.6	32.4
①+②+⑤+⑥	22.4	34.7	15.3	31.3	**24.3**	91.7	19.0	45.3	**23.2**	**46.0**

Tabelle 67: Sequence-Tagging Ergebnisse für Datensatz **Both** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	26.1	51.6	21.8	39.0	23.6	73.9	21.1	49.3	14.6	19.9
Grammatical ②	28.6	50.3	25.0	43.1	22.2	90.0	27.9	90.0	18.8	26.2
LDA Wikipedia ③	16.4	37.8	2.8	3.2	2.9	1.3	24.9	86.6	15.7	45.2
LDA Dataset ④	22.0	40.8	3.1	2.7	4.2	4.5	6.1	8.8	8.8	9.2
Character n-grams	25.9	50.9	25.5	46.0	15.9	38.4	24.9	**90.8**	20.4	44.5
Word Embeddings ⑤	26.0	52.1	14.3	23.3	21.9	90.5	23.0	57.7	21.8	38.7
Character Embeddings ⑥	28.0	51.6	23.6	44.1	23.2	**90.9**	21.4	48.2	22.0	28.9
①+②	28.2	52.2	**26.3**	43.1	**25.2**	83.7	27.0	76.7	15.1	23.2
①+③	16.2	23.1	25.6	47.2	24.9	77.1	20.7	47.0	**24.9**	39.5
①+④	28.2	45.1	25.7	45.3	24.5	77.9	24.1	61.3	16.6	21.5
①+⑤	26.9	52.6	24.9	**49.7**	19.6	56.8	26.7	87.3	21.9	40.2
①+⑥	28.9	52.2	26.1	45.0	19.0	53.2	**29.3**	89.5	21.5	42.3
①+②+⑤	27.3	53.2	16.1	28.6	20.2	58.4	27.2	86.8	21.2	32.2
①+②+⑥	**29.5**	51.8	26.2	49.3	18.1	51.6	19.3	38.4	21.6	**47.0**
①+②+⑤+⑥	28.6	**53.2**	26.2	46.4	16.9	44.4	20.0	40.9	23.8	36.6

Tabelle 68: Sequence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	19.2	50.8	16.1	**73.1**	21.5	87.0	25.7	71.6	20.8	35.9
Grammatical ②	24.8	35.5	13.3	28.2	13.5	34.1	27.0	77.1	16.2	22.4
LDA Wikipedia ③	18.8	46.7	1.8	1.3	3.6	4.8	1.8	3.4	4.7	4.5
LDA Dataset ④	7.9	8.8	2.4	1.0	4.1	5.6	4.4	4.7	14.7	**38.5**
Character n-grams	31.7	53.1	12.8	26.0	19.6	92.0	22.8	49.9	**22.6**	33.0
Word Embeddings ⑤	22.6	32.0	17.5	52.4	20.5	76.0	27.1	79.7	19.4	26.8
Character Embeddings ⑥	31.2	47.1	**18.1**	48.9	21.8	88.1	27.6	70.5	20.7	30.2
①+②	**31.7**	57.0	15.9	72.7	20.7	91.2	25.3	68.8	22.0	37.8
①+③	29.9	56.7	17.7	65.4	20.2	**92.4**	25.7	71.8	19.5	31.6
①+④	29.9	56.7	16.5	54.0	21.0	89.2	25.5	71.7	19.5	31.1
①+⑤	19.2	50.8	17.6	69.6	20.7	91.1	27.3	83.2	20.9	37.2
①+⑥	30.1	48.1	17.3	54.6	21.7	85.8	27.8	74.8	21.3	34.2
①+②+⑤	30.9	**57.5**	16.6	57.3	21.0	90.0	27.5	83.7	17.6	36.9
①+②+⑥	30.5	47.1	17.3	55.0	21.6	85.9	27.9	75.0	20.9	34.3
①+②+⑤+⑥	30.5	48.5	17.7	63.0	**22.0**	88.8	**29.7**	**86.1**	21.9	37.7

Tabelle 69: Sequence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	30.4	55.7	17.8	60.2	23.3	91.7	27.6	82.5	22.2	46.4
Grammatical ②	25.4	36.6	16.2	27.1	14.1	34.5	26.3	81.3	18.3	23.8
LDA Wikipedia ③	5.4	5.8	2.4	2.0	2.7	2.3	1.5	2.4	4.3	4.0
LDA Dataset ④	9.5	11.8	4.2	3.1	4.1	4.3	7.1	10.4	13.7	25.1
Character n-grams	23.1	55.4	20.2	49.1	22.4	91.7	20.1	44.7	19.5	26.0
Word Embeddings ⑤	28.6	45.2	22.3	47.8	22.6	71.1	25.9	75.1	18.6	22.4
Character Embeddings ⑥	29.3	43.4	19.4	33.1	19.9	58.8	27.3	76.9	22.0	30.3
①+②	30.6	56.0	17.5	60.4	23.4	**91.7**	23.8	64.1	22.2	36.4
①+③	30.4	55.7	22.1	54.2	-	-	27.6	82.6	22.2	**46.4**
①+④	**30.7**	55.8	22.1	54.2	23.3	91.7	**27.7**	82.7	23.2	41.2
①+⑤	21.1	55.2	17.5	**60.4**	25.4	91.2	25.7	**93.1**	24.1	42.8
①+⑥	28.9	42.9	21.4	46.7	26.0	87.5	25.8	71.9	25.3	37.7
①+②+⑤	30.6	56.2	22.7	55.1	23.3	91.7	27.4	90.4	23.5	36.9
①+②+⑥	28.8	42.7	21.4	45.5	**26.1**	87.6	26.0	73.6	**25.4**	38.2
①+②+⑤+⑥	30.6	**56.2**	**23.7**	55.8	26.1	87.6	26.7	76.2	25.3	38.2

Tabelle 70: Sequence-Tagging Ergebnisse für Datensatz **Both** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	27.5	50.9	14.8	**48.1**	20.6	91.0	25.1	66.2	13.0	48.3
Grammatical ②	23.8	32.9	20.1	29.0	15.9	39.6	24.8	62.3	19.7	26.7
LDA Wikipedia ③	18.6	39.4	2.5	2.7	20.4	82.7	24.8	82.7	4.7	4.7
LDA Dataset ④	19.7	37.5	3.6	2.7	3.9	3.7	23.0	80.4	7.2	6.9
Character n-grams	21.3	52.9	22.5	34.7	20.8	**91.0**	25.9	63.8	23.1	32.4
Word Embeddings ⑤	23.6	33.3	19.3	27.9	24.9	90.3	25.0	69.8	20.3	28.1
Character Embeddings ⑥	26.3	37.1	22.8	33.0	19.8	54.3	26.7	71.6	21.2	27.5
①+②	28.3	50.9	23.3	47.5	26.1	90.5	25.8	68.2	22.7	39.4
①+③	18.3	53.2	24.5	43.7	20.6	91.0	25.1	66.2	13.0	**48.3**
①+④	27.4	50.9	21.9	47.2	26.1	90.3	25.2	63.4	23.3	44.0
①+⑤	18.3	**53.2**	22.1	47.5	26.2	86.2	25.2	66.4	23.2	43.9
①+⑥	26.4	37.9	24.2	39.1	23.9	69.0	**28.3**	84.3	24.5	36.3
①+②+⑤	**29.9**	47.3	22.2	47.9	**26.5**	90.5	25.6	67.9	22.3	39.9
①+②+⑥	26.5	38.0	24.7	39.8	24.1	69.4	27.7	84.3	**24.5**	36.4
①+②+⑤+⑥	26.8	39.1	**24.7**	39.9	25.6	86.3	28.0	**84.3**	24.5	36.4

Tabelle 71: Sequence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	30.0	48.1	15.8	54.3	21.1	87.6	24.0	65.6	21.0	33.2
Grammatical ②	24.9	35.7	10.7	18.1	11.7	27.7	17.2	34.0	14.9	20.3
LDA Wikipedia ③	18.2	**48.8**	2.3	7.3	3.6	4.7	24.6	**88.1**	12.5	31.7
LDA Dataset ④	6.5	6.7	13.2	**59.1**	19.1	76.0	4.7	6.4	14.6	**39.3**
Character n-grams	27.4	40.4	10.9	20.3	-	-	20.4	42.1	17.3	23.2
Word Embeddings ⑤	24.3	36.8	10.4	15.5	14.7	40.5	18.8	42.5	16.8	22.9
Character Embeddings ⑥	**30.9**	47.2	12.8	23.5	16.5	47.6	19.4	40.6	19.2	25.8
①+②	29.7	47.2	15.8	44.1	21.3	87.8	24.8	67.2	**21.9**	34.9
①+③	27.5	44.7	15.3	42.7	20.9	87.7	24.1	65.7	19.2	33.2
①+④	27.5	44.7	15.8	50.1	21.1	87.7	24.1	65.7	19.5	32.8
①+⑤	27.4	44.6	15.8	54.3	20.7	89.1	24.3	67.8	20.6	34.5
①+⑥	29.6	47.3	16.8	49.3	21.3	89.1	26.7	76.6	20.3	32.2
①+②+⑤	28.0	45.3	16.0	55.1	21.3	**89.3**	25.6	73.9	20.8	34.5
①+②+⑥	30.0	46.2	17.0	50.0	**21.7**	89.2	26.8	77.1	20.7	32.5
①+②+⑤+⑥	29.9	46.2	**17.0**	56.5	21.5	89.3	**26.9**	77.2	21.8	34.9

Tabelle 72: Sequence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	26.3	41.0	18.3	43.3	23.3	85.4	23.4	64.2	20.9	33.7
Grammatical ②	-	-	13.4	19.7	-	-	-	-	17.4	23.4
LDA Wikipedia ③	5.1	5.4	2.1	1.7	-	-	24.6	**84.8**	4.2	3.8
LDA Dataset ④	8.1	8.8	13.4	**47.1**	19.2	68.2	4.5	6.8	13.7	25.5
Character n-grams	25.7	38.1	15.1	25.7	-	-	21.4	51.1	18.6	24.2
Word Embeddings ⑤	23.4	33.6	17.5	30.4	13.5	30.7	20.9	54.7	18.1	21.5
Character Embeddings ⑥	**29.1**	43.8	19.9	34.6	13.2	30.5	**27.1**	42.1	20.9	25.8
①+②	26.0	38.3	19.2	39.7	23.6	85.1	24.1	65.9	21.1	33.6
①+③	25.9	38.5	18.5	38.5	23.5	82.9	23.1	62.2	21.5	34.2
①+④	26.5	41.2	18.4	43.5	23.9	85.5	23.5	64.5	21.7	34.6
①+⑤	26.4	41.2	18.9	39.1	23.8	**86.9**	23.5	64.5	22.6	36.5
①+⑥	28.4	44.0	19.8	41.5	23.0	71.2	25.6	73.8	23.6	34.1
①+②+⑤	26.8	39.6	19.7	40.3	24.1	85.9	24.2	66.1	22.3	34.5
①+②+⑥	28.6	44.2	20.3	41.9	**25.2**	83.2	26.0	72.1	24.0	34.6
①+②+⑤+⑥	28.6	**44.3**	**20.6**	42.2	25.2	82.9	26.0	72.2	**24.6**	**37.0**

Tabelle 73: Sequence-Tagging Ergebnisse für Datensatz **Both** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	25.8	38.9	20.8	35.7	24.0	79.1	25.2	64.2	**24.7**	**37.9**
Grammatical ②	20.9	32.4	16.2	21.9	12.4	25.8	-	-	17.7	22.9
LDA Wikipedia ③	17.6	38.7	-	-	20.5	83.0	24.2	**86.8**	4.5	4.5
LDA Dataset ④	18.6	37.4	3.9	4.4	18.6	71.2	5.2	6.5	12.0	19.5
Character n-grams	24.4	36.5	18.2	28.8	11.7	23.0	19.2	37.4	21.3	26.1
Word Embeddings ⑤	23.6	33.5	18.5	28.0	15.6	37.8	19.0	39.9	19.7	24.6
Character Embeddings ⑥	26.3	38.5	21.5	31.4	17.0	40.5	22.4	46.7	21.1	27.0
①+②	26.9	40.0	22.0	37.2	25.0	84.9	25.5	64.1	21.6	35.6
①+③	25.8	38.9	20.7	35.6	24.1	82.7	25.2	64.4	22.1	34.8
①+④	25.7	38.8	20.8	37.1	24.1	82.7	25.2	64.2	21.5	36.5
①+⑤	25.8	38.9	21.1	36.1	24.5	81.4	25.1	64.3	21.2	35.8
①+⑥	26.3	38.9	22.6	37.9	25.5	81.8	26.3	71.1	23.8	35.1
①+②+⑤	26.2	39.5	22.4	37.6	25.1	**85.0**	25.7	64.6	21.3	36.5
①+②+⑥	27.0	40.4	23.0	38.1	**26.0**	82.4	26.7	68.4	23.5	34.9
①+②+⑤+⑥	**27.2**	**40.7**	**23.2**	**38.4**	25.8	81.7	**26.8**	68.5	23.5	35.0

Tabelle 74: Sequence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

A.2.2 Klassisches Machine Learning: Cross-Platform

In diesem Kapitel werden die einzelnen Cross-Platform Ergebnisse der klassischen Machine Learning Verfahren aufgeführt. Es folgen Tabellen für jeden Klassifikator und Datensatz sowie zusammenfassende Abbildungen. In allen Tabellen sowie Abbildungen wird jeweils der Datensatz des verwendeten Trainingssets angegeben. Der Datensatz des verwendeten Testsets ist für Cross-Platform Untersuchungen immer der jeweils andere Datensatz. Der kombinierte Datensatz Both wurde hierbei nicht untersucht.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	29.3	47.9	14.3	25.5	9.1	23.7	19.0	37.8	17.1	27.9
Grammatical ②	27.9	50.7	13.2	21.9	15.3	46.6	14.5	26.8	**20.0**	30.1
LDA Wikipedia ③	19.4	**51.0**	2.6	3.7	2.1	2.0	4.9	7.1	5.0	6.4
LDA Dataset ④	6.0	6.9	2.3	2.1	3.1	3.3	6.0	11.4	9.2	18.1
Character n-grams	**29.7**	48.7	11.6	**44.5**	19.0	**90.9**	**26.4**	**90.3**	17.8	**36.9**
Word Embeddings ⑤	19.6	27.5	13.6	20.3	**19.2**	90.4	18.6	40.4	19.6	26.1
Character Embeddings ⑥	25.6	41.1	13.4	18.4	16.5	54.2	20.2	44.0	18.7	26.8
①+②	25.7	39.7	**18.2**	32.1	8.9	17.6	15.2	27.1	18.9	28.3
①+③	24.3	33.4	7.9	12.1	17.8	65.4	5.3	6.3	19.2	27.7
①+④	28.5	49.0	13.7	24.6	12.4	32.4	22.5	56.9	17.1	24.2
①+⑤	27.0	48.5	12.3	18.4	17.2	64.0	6.8	8.3	19.7	24.4
①+⑥	24.4	34.8	14.7	23.6	15.8	55.7	20.3	41.0	17.6	22.9
①+②+⑤	23.0	31.4	16.7	27.7	17.9	62.4	13.2	22.6	19.4	25.7
①+②+⑥	24.8	35.6	16.2	25.6	19.0	66.2	16.5	29.4	20.0	25.2
①+②+⑤+⑥	24.3	33.8	16.8	27.1	16.3	50.7	15.4	25.8	18.9	23.6

Tabelle 75: Cross Platform Sequence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	25.7	51.4	12.8	36.3	19.5	79.6	20.0	50.4	13.9	18.8
Grammatical ②	26.2	50.6	14.0	41.5	19.4	92.0	24.4	91.9	17.5	24.7
LDA Wikipedia ③	17.4	42.5	1.9	1.3	1.8	1.1	24.9	88.7	12.4	30.0
LDA Dataset ④	20.6	44.7	1.2	0.9	2.1	2.7	5.2	7.7	5.0	4.9
Character n-grams	25.1	50.0	**15.5**	49.5	13.2	38.4	24.0	**92.3**	17.5	36.2
Word Embeddings ⑤	25.8	52.4	8.8	17.1	19.3	**92.9**	23.7	65.9	19.4	33.3
Character Embeddings ⑥	27.0	51.4	14.2	46.9	19.4	92.8	23.0	54.8	19.4	30.1
①+②	26.4	51.3	14.3	39.8	**20.0**	84.6	25.4	77.0	13.9	21.3
①+③	14.8	22.7	13.7	51.9	19.4	79.2	19.4	47.4	**20.9**	34.9
①+④	26.7	44.9	13.8	47.5	19.6	80.6	22.4	61.8	14.2	19.6
①+⑤	26.0	52.3	13.2	53.3	16.9	61.9	25.6	90.3	18.9	34.9
①+⑥	27.7	51.7	14.7	44.9	16.1	55.6	**29.0**	90.7	18.1	34.7
①+②+⑤	26.2	52.4	10.4	28.6	16.4	57.1	26.0	89.5	19.3	30.2
①+②+⑥	**28.0**	51.1	15.0	**56.3**	16.1	55.8	19.4	43.8	16.3	**40.9**
①+②+⑤+⑥	26.8	**52.8**	14.5	44.3	14.8	46.7	19.4	43.5	19.9	33.2

Tabelle 76: Cross Platform Sequence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	17.3	53.0	10.5	**43.3**	19.7	88.5	25.4	73.2	19.6	39.8
Grammatical ②	24.5	33.8	**16.8**	29.0	13.8	37.6	26.1	76.2	16.8	20.3
LDA Wikipedia ③	19.2	49.6	2.3	2.6	3.2	4.9	1.4	2.4	4.9	5.9
LDA Dataset ④	6.0	6.9	2.4	2.1	3.4	3.7	3.5	3.1	9.1	18.2
Character n-grams	**28.2**	48.0	15.2	26.5	19.6	89.8	20.7	45.0	19.5	28.5
Word Embeddings ⑤	20.3	26.6	14.4	34.7	**20.4**	79.5	25.5	79.9	19.8	26.1
Character Embeddings ⑥	27.3	39.3	16.0	37.1	19.5	74.2	27.8	72.5	19.9	26.3
①+②	28.0	51.1	10.4	42.9	19.2	90.1	25.2	72.4	**20.4**	38.2
①+③	26.7	50.4	12.5	42.0	19.1	**90.7**	25.3	73.3	18.7	31.3
①+④	26.2	50.2	13.1	39.8	19.3	90.1	25.3	73.1	18.5	31.5
①+⑤	17.3	**53.0**	12.8	40.5	19.3	90.2	26.0	84.6	19.3	37.5
①+⑥	27.8	43.9	15.3	36.0	20.2	87.8	26.8	76.2	19.1	34.3
①+②+⑤	27.2	50.5	12.9	40.7	19.4	89.7	26.1	84.9	16.9	**43.3**
①+②+⑥	27.3	40.8	15.2	36.1	20.2	87.8	27.1	75.9	19.2	33.9
①+②+⑤+⑥	28.1	44.3	15.3	36.1	20.1	83.3	**28.2**	**87.5**	19.6	37.5

Tabelle 77: Cross Platform Sequence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	25.8	48.9	12.3	**75.9**	19.3	93.1	23.9	70.1	9.8	32.5
Grammatical ②	22.9	33.7	11.1	23.6	13.9	41.0	24.3	63.6	**20.1**	30.0
LDA Wikipedia ③	19.2	43.8	1.8	1.5	19.5	84.9	24.4	84.7	4.5	4.8
LDA Dataset ④	18.6	42.7	1.7	0.7	1.9	1.4	23.6	**85.7**	3.5	3.9
Character n-grams	18.8	**50.1**	12.6	29.4	19.3	93.1	24.2	64.4	19.6	29.5
Word Embeddings ⑤	23.1	33.7	12.0	24.5	19.5	92.7	23.8	72.3	17.9	26.0
Character Embeddings ⑥	25.5	37.1	13.4	27.7	16.9	58.1	26.0	74.7	19.3	28.3
①+②	26.1	47.7	**14.7**	53.4	19.3	92.5	24.4	72.3	17.5	32.4
①+③	16.6	49.7	13.9	44.2	19.3	**93.1**	23.9	70.3	9.8	32.5
①+④	25.8	48.9	14.0	54.6	19.4	92.5	23.8	70.0	16.0	32.7
①+⑤	16.6	49.7	14.0	54.7	19.6	90.2	23.8	70.1	16.2	**32.9**
①+⑥	25.9	39.2	14.1	39.4	19.2	75.1	25.9	85.2	19.8	30.1
①+②+⑤	**27.7**	46.5	14.0	54.0	19.7	92.7	24.4	72.0	18.2	32.4
①+②+⑥	26.2	39.5	14.3	39.7	19.1	75.3	26.0	85.2	19.8	30.4
①+②+⑤+⑥	26.2	39.5	14.3	39.9	**20.0**	90.1	**26.1**	85.2	19.8	30.4

Tabelle 78: Cross Platform Sequence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	27.4	45.3	12.1	39.2	19.5	88.2	24.4	71.0	17.4	31.5
Grammatical ②	24.6	34.0	15.2	23.0	12.5	31.9	16.8	32.2	17.7	21.6
LDA Wikipedia ③	18.9	**52.1**	6.1	24.5	3.1	4.7	24.7	**87.9**	14.5	**45.5**
LDA Dataset ④	5.9	6.4	10.2	39.7	19.6	80.1	4.9	7.6	8.9	18.1
Character n-grams	24.1	35.3	13.8	22.5	-	-	19.2	38.7	18.2	23.2
Word Embeddings ⑤	21.3	29.1	13.6	19.9	15.6	46.2	17.3	35.1	18.8	22.4
Character Embeddings ⑥	27.0	39.1	**16.1**	26.0	17.7	53.9	17.0	32.5	18.5	22.3
①+②	27.4	44.2	13.8	35.3	19.8	88.7	24.8	71.1	18.0	36.2
①+③	25.3	42.6	13.0	33.9	19.5	88.1	24.5	71.2	18.2	33.4
①+④	25.3	42.7	13.0	33.9	19.5	88.2	24.5	71.2	18.9	32.8
①+⑤	25.5	42.5	12.1	39.2	19.4	89.0	24.5	72.4	16.9	34.4
①+⑥	27.5	45.0	13.7	37.7	20.1	88.5	**27.3**	82.3	19.0	26.5
①+②+⑤	25.8	42.9	12.7	**40.2**	19.4	**89.3**	25.5	79.3	17.4	32.4
①+②+⑥	27.3	42.3	14.2	38.5	**21.7**	82.6	27.2	82.7	19.0	30.7
①+②+⑤+⑥	**27.7**	45.0	14.3	38.6	21.6	82.7	27.1	82.3	**19.1**	30.8

Tabelle 79: Cross Platform Sequence-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	25.0	42.4	12.9	39.3	19.4	86.2	24.0	72.1	17.9	28.9
Grammatical ②	21.2	35.7	9.2	16.1	10.6	27.3	-	-	17.6	25.1
LDA Wikipedia ③	18.0	**43.3**	-	-	19.5	85.2	24.1	**88.5**	4.2	4.6
LDA Dataset ④	18.2	43.1	2.0	2.1	18.7	80.3	3.9	5.6	14.7	**43.0**
Character n-grams	22.7	34.0	11.2	26.2	10.1	25.1	18.2	37.6	16.8	22.9
Word Embeddings ⑤	23.8	36.0	12.3	27.2	15.0	45.5	19.9	47.8	16.8	23.8
Character Embeddings ⑥	**27.2**	40.3	13.3	27.5	15.0	45.8	22.8	51.1	18.9	27.3
①+②	26.4	42.9	13.2	38.5	19.7	89.6	24.1	71.2	18.7	31.7
①+③	25.1	42.4	12.9	39.4	19.4	88.9	24.1	72.3	16.2	27.6
①+④	25.0	42.4	12.8	39.3	19.4	88.8	24.0	72.2	17.8	31.9
①+⑤	25.0	42.3	13.0	39.7	19.5	87.7	24.1	72.1	18.0	30.5
①+⑥	26.2	40.8	13.5	40.2	19.7	87.1	**26.0**	79.5	19.4	29.8
①+②+⑤	25.5	42.0	13.5	39.4	19.7	89.7	24.3	71.3	18.7	31.1
①+②+⑥	26.3	43.2	13.7	40.4	19.9	87.6	25.8	74.5	19.5	29.9
①+②+⑤+⑥	26.4	43.3	**13.9**	**40.8**	**20.0**	**89.8**	25.7	74.5	**19.7**	31.4

Tabelle 80: Cross Platform Sequence-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

A.3 Document-Tagging

In diesem Kapitel werden die detaillierten Ergebnisse des Document Taggings aufgeführt.

A.3.1 Klassisches Machine Learning

Es folgen die detaillierten Document Tagging Ergebnisse für die Klassifikatoren Random Forest (RF), Support Vector Machine (SVM) und Support Vector Machine mit linearem Kernel (SVM-Linear) sowie Conditional Random Fields (CRF).

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	76.1	86.0	37.8	75.5	60.8	66.0	61.7	70.3	47.3	75.5
Grammatical ②	72.9	86.2	44.1	72.6	59.3	66.7	59.1	66.1	49.3	71.1
LDA Wikipedia ③	71.5	85.7	40.8	74.0	**63.4**	71.6	56.7	65.9	47.0	62.1
LDA Dataset ④	71.8	87.3	37.5	75.5	56.0	66.0	52.5	65.3	59.0	71.9
Character n-grams	74.2	86.7	40.1	75.2	54.4	66.0	59.7	70.5	45.1	70.8
Word Embeddings ⑤	77.7	89.1	43.7	74.2	61.4	66.7	65.5	71.2	57.7	75.9
Character Embeddings ⑥	72.4	87.5	45.2	**79.1**	54.8	61.7	64.4	70.6	51.2	72.7
①+②	76.1	86.9	35.8	73.1	54.7	56.7	57.6	69.8	42.4	73.5
①+③	73.6	86.2	39.8	71.8	62.4	**73.0**	60.7	65.3	49.5	70.0
①+④	75.2	88.0	36.9	75.8	60.4	68.1	59.8	69.6	57.7	67.2
①+⑤	78.7	88.9	41.9	75.2	57.9	67.4	**66.8**	**73.7**	57.2	74.3
①+⑥	**80.5**	**90.2**	42.7	78.1	56.9	65.2	64.8	71.3	55.8	71.5
①+②+⑤	78.9	89.6	42.0	79.1	56.0	66.0	66.3	72.4	53.7	71.5
①+②+⑥	76.4	88.4	**46.9**	78.1	61.0	68.8	62.5	70.8	53.1	64.0
①+②+⑤+⑥	79.0	90.0	43.7	77.8	58.0	66.7	64.8	71.9	**59.1**	**76.7**

Tabelle 81: Document-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	64.7	87.1	41.4	64.0	66.7	68.8	61.0	71.6	58.8	59.0
Grammatical ②	64.6	85.3	50.9	65.8	68.4	71.0	57.2	70.1	61.4	61.9
LDA Wikipedia ③	62.3	75.9	41.2	62.9	64.4	66.2	57.9	67.1	56.4	56.5
LDA Dataset ④	65.3	80.2	45.4	63.1	66.7	68.8	56.0	66.1	59.2	60.1
Character n-grams	59.4	87.5	48.8	65.2	65.9	66.2	60.0	71.7	60.0	60.8
Word Embeddings ⑤	67.8	87.4	55.3	65.9	**71.5**	**72.8**	64.3	69.5	70.5	70.6
Character Embeddings ⑥	68.2	88.1	55.9	66.0	71.2	71.8	62.2	67.2	67.5	67.7
①+②	64.3	87.4	51.3	66.9	67.8	69.2	59.3	71.0	57.9	57.9
①+③	65.5	86.2	49.7	67.0	66.1	67.0	60.8	70.2	63.9	64.2
①+④	66.8	86.1	49.4	65.9	69.0	69.8	64.1	70.1	64.0	64.0
①+⑤	69.1	88.4	56.1	66.3	70.0	71.0	**65.3**	**73.1**	**71.0**	**71.5**
①+⑥	71.2	88.6	50.6	65.5	70.3	71.0	62.8	71.5	67.8	68.1
①+②+⑤	70.4	88.7	56.7	68.1	70.2	71.6	65.2	70.8	68.6	69.0
①+②+⑥	**71.4**	89.1	52.5	66.7	71.3	71.8	62.2	70.8	67.0	67.2
①+②+⑤+⑥	70.8	**89.7**	**57.7**	**68.1**	69.3	69.8	64.8	71.5	69.7	70.4

Tabelle 82: Document-Tagging Ergebnisse für Datensatz **Both** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	58.3	88.3	40.7	55.7	63.4	68.1	62.9	**72.7**	58.3	63.3
Grammatical ②	58.0	87.5	56.5	62.6	58.2	68.3	58.4	69.9	50.8	57.8
LDA Wikipedia ③	56.9	81.3	43.9	53.8	58.2	64.4	57.2	65.5	50.9	56.2
LDA Dataset ④	60.6	79.5	45.2	53.6	63.7	69.5	57.3	68.4	50.9	53.9
Character n-grams	55.5	89.1	51.2	59.2	59.9	65.8	56.3	71.2	45.0	56.8
Word Embeddings ⑤	55.5	89.7	**58.6**	63.3	63.4	70.9	63.4	69.9	57.7	61.4
Character Embeddings ⑥	63.5	90.4	54.7	61.6	63.9	72.0	62.2	69.5	55.3	60.1
①+②	55.7	88.7	54.2	62.2	57.6	66.1	62.1	71.6	52.1	58.1
①+③	62.6	83.2	49.6	59.3	63.3	68.9	63.5	72.4	56.4	62.3
①+④	59.5	86.3	47.9	60.3	65.1	**72.8**	61.7	71.0	51.7	59.4
①+⑤	56.7	89.7	58.3	63.7	65.0	70.3	62.4	71.9	58.5	63.6
①+⑥	61.5	90.2	57.7	**64.4**	62.3	66.4	62.8	70.5	59.3	64.6
①+②+⑤	56.6	89.6	58.4	63.6	62.3	67.5	63.4	70.3	54.2	62.0
①+②+⑥	62.2	90.4	57.2	63.6	61.8	69.7	62.7	70.8	56.5	64.6
①+②+⑤+⑥	**64.4**	**90.7**	58.5	63.9	**66.5**	72.3	**65.0**	72.5	**61.3**	**64.9**

Tabelle 83: Document-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	53.0	85.1	35.6	80.3	61.4	66.7	58.6	68.7	57.7	75.9
Grammatical ②	75.0	85.7	40.8	56.9	52.9	68.1	58.8	62.9	42.6	74.3
LDA Wikipedia ③	61.5	75.6	7.7	6.4	57.1	63.1	24.3	32.1	48.8	71.9
LDA Dataset ④	62.9	78.5	38.8	66.6	47.9	63.8	53.1	59.3	50.7	61.3
Character n-grams	66.5	86.4	39.8	73.3	54.9	**70.9**	53.5	61.5	50.5	71.5
Word Embeddings ⑤	65.1	87.3	**45.7**	61.0	53.5	66.0	**64.7**	66.3	58.2	66.8
Character Embeddings ⑥	**77.9**	**88.4**	41.5	64.4	52.2	68.8	62.8	64.3	65.8	75.1
①+②	66.4	86.9	35.9	**80.6**	55.0	64.5	62.8	**70.5**	55.5	76.3
①+③	68.0	87.2	41.3	79.7	55.5	70.2	58.4	68.6	61.7	73.9
①+④	67.7	87.2	42.4	72.3	61.4	66.7	61.9	69.7	**68.3**	**77.9**
①+⑤	53.0	85.1	43.5	73.9	58.1	66.0	60.7	68.6	55.7	75.5
①+⑥	77.5	87.1	43.5	74.3	58.7	64.5	60.6	68.0	66.1	75.9
①+②+⑤	69.6	87.3	44.7	74.5	**64.3**	69.5	61.5	69.5	44.1	74.3
①+②+⑥	76.4	87.3	43.3	74.2	59.3	65.2	60.6	68.0	61.2	73.9
①+②+⑤+⑥	77.8	87.5	44.0	76.2	64.3	69.5	62.1	69.8	65.5	76.7

Tabelle 84: Document-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
*F*1-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	62.7	86.1	42.5	68.1	39.6	63.0	55.0	71.4	51.9	58.3
Grammatical ②	65.8	81.5	53.0	60.3	62.6	62.8	55.9	62.9	62.2	63.8
LDA Wikipedia ③	60.0	81.8	39.6	62.7	66.5	67.4	22.6	29.2	55.4	55.6
LDA Dataset ④	60.5	72.7	42.4	57.6	62.1	63.4	54.4	58.5	59.5	60.2
Character n-grams	50.4	86.8	45.4	67.7	39.6	63.0	57.4	65.1	59.5	59.9
Word Embeddings ⑤	**71.2**	85.7	**56.8**	65.4	64.1	64.2	62.3	64.9	**70.9**	**72.0**
Character Embeddings ⑥	71.1	85.7	55.4	63.7	67.7	68.8	62.5	66.6	70.2	71.7
①+②	61.7	86.4	43.3	68.5	39.6	63.0	63.6	**71.5**	69.6	69.9
①+③	62.7	86.1	50.7	68.3	39.6	63.0	54.9	71.3	51.9	58.3
①+④	62.4	86.1	50.9	68.4	39.6	63.0	55.2	71.4	66.5	67.7
①+⑤	51.1	**87.3**	43.7	68.6	70.5	71.6	43.1	71.2	68.2	68.3
①+⑥	69.5	82.8	53.1	63.7	**71.6**	**73.4**	**63.7**	71.4	66.6	67.2
①+②+⑤	68.1	86.1	51.4	**68.7**	39.6	63.0	61.1	69.7	68.1	68.3
①+②+⑥	69.6	82.9	53.5	65.0	71.3	73.0	62.9	70.5	66.4	67.0
①+②+⑤+⑥	69.6	82.9	54.0	65.4	71.6	72.6	62.9	70.5	66.6	66.7

Tabelle 85: Document-Tagging Ergebnisse für Datensatz **Both** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	57.4	88.4	26.7	56.4	43.5	73.9	62.6	70.5	38.6	63.0
Grammatical ②	60.2	80.7	57.1	61.1	54.8	58.8	54.9	59.5	52.1	54.2
LDA Wikipedia ③	56.1	81.0	43.7	51.1	57.7	65.3	55.4	64.3	44.4	51.6
LDA Dataset ④	56.1	78.1	41.9	45.5	56.6	62.2	52.0	62.1	51.8	52.3
Character n-grams	47.0	88.5	44.2	58.3	43.5	73.9	56.3	67.0	53.8	60.7
Word Embeddings ⑤	**68.9**	86.7	**58.9**	**64.4**	64.4	70.0	58.2	62.9	59.1	59.1
Character Embeddings ⑥	67.9	87.4	58.0	62.0	67.5	70.6	58.1	61.5	**65.8**	**66.9**
①+②	57.3	88.3	42.0	60.0	62.9	75.4	62.5	70.3	61.9	65.6
①+③	50.3	**89.1**	52.9	62.1	43.5	73.9	62.7	70.7	38.6	63.0
①+④	57.3	88.3	43.6	61.1	62.7	74.8	62.7	70.5	57.4	66.6
①+⑤	48.7	89.0	43.4	60.8	**67.8**	**76.5**	62.8	70.6	56.8	66.2
①+⑥	67.8	86.0	54.2	58.8	62.9	71.7	62.8	70.8	63.2	65.6
①+②+⑤	63.2	87.3	43.7	61.0	67.6	75.6	63.1	70.8	60.6	64.3
①+②+⑥	68.0	86.2	54.6	59.6	63.4	72.0	63.2	71.2	62.9	65.3
①+②+⑤+⑥	68.1	86.3	54.6	59.6	67.3	76.2	**63.4**	**71.3**	62.9	65.3

Tabelle 86: Document-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	77.8	86.9	43.4	73.1	63.8	69.5	**63.8**	**71.8**	64.3	74.3
Grammatical ②	75.2	86.6	40.1	54.7	51.0	68.8	40.4	67.9	42.6	74.3
LDA Wikipedia ③	49.3	53.4	3.8	6.1	57.1	63.8	26.3	33.2	55.1	58.1
LDA Dataset ④	56.5	69.5	30.9	44.3	61.3	64.5	48.0	48.4	48.5	52.6
Character n-grams	66.3	74.7	34.5	46.4	62.8	69.5	58.3	64.5	46.0	54.9
Word Embeddings ⑤	62.9	87.1	41.2	64.4	56.6	66.7	24.3	32.1	60.0	69.2
Character Embeddings ⑥	74.8	88.0	41.6	60.7	57.5	66.0	60.5	61.9	**67.2**	76.3
①+②	**78.9**	**88.0**	42.5	72.7	63.8	69.5	63.3	71.4	61.2	73.9
①+③	75.7	85.7	41.3	71.3	63.8	69.5	63.4	71.4	63.6	73.5
①+④	77.2	86.8	42.3	72.4	63.8	69.5	63.7	71.7	64.3	74.3
①+⑤	77.0	86.1	42.9	73.3	63.2	68.8	60.4	69.0	66.1	75.9
①+⑥	75.7	86.2	44.0	73.7	63.8	69.5	61.4	67.9	61.5	74.3
①+②+⑤	75.9	86.6	44.2	74.2	**66.1**	**71.6**	63.3	70.7	66.4	76.3
①+②+⑥	77.0	86.7	44.1	74.0	58.4	59.6	61.9	69.6	61.5	74.3
①+②+⑤+⑥	76.8	87.1	**45.6**	**75.0**	66.1	71.6	62.0	69.6	65.9	**77.1**

Tabelle 87: Document-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	66.9	84.1	52.1	62.8	69.5	71.6	65.8	73.9	64.6	64.7
Grammatical ②	61.4	77.5	54.6	62.5	61.9	62.2	55.9	58.6	60.2	60.6
LDA Wikipedia ③	56.6	70.4	38.9	45.4	65.1	66.0	56.5	58.8	59.5	59.9
LDA Dataset ④	52.6	65.4	37.8	42.5	59.3	59.8	49.3	51.9	61.3	62.2
Character n-grams	55.7	64.7	42.6	45.7	62.0	63.2	60.8	68.0	59.8	59.9
Word Embeddings ⑤	70.9	86.3	**56.8**	**69.4**	70.1	70.2	41.4	70.8	67.2	68.3
Character Embeddings ⑥	**71.4**	**86.5**	53.9	60.8	**72.5**	**72.8**	**69.2**	**85.3**	**71.7**	**72.5**
①+②	69.8	84.5	54.5	64.4	69.5	71.6	64.5	73.0	66.2	66.3
①+③	69.6	83.3	51.9	61.7	69.0	71.2	65.5	73.4	64.4	64.5
①+④	67.6	84.5	52.8	63.3	69.4	71.6	65.8	73.9	65.2	65.2
①+⑤	69.6	84.2	52.8	63.7	68.8	71.0	65.9	73.9	64.8	64.9
①+⑥	68.4	85.3	52.9	62.3	67.9	68.6	63.4	70.6	66.2	66.8
①+②+⑤	70.4	85.8	54.7	64.5	70.4	72.2	64.8	73.1	65.8	66.0
①+②+⑥	67.2	85.3	54.4	64.5	70.6	72.4	64.8	73.1	66.4	67.0
①+②+⑤+⑥	70.1	86.0	54.6	64.6	70.5	72.4	65.2	73.5	66.4	67.0

Tabelle 88: Document-Tagging Ergebnisse für Datensatz **Both** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	62.6	84.9	52.7	57.8	66.8	75.9	63.0	72.0	58.5	62.3
Grammatical ②	10.1	11.2	**57.6**	61.3	56.6	60.8	53.8	57.9	50.9	51.0
LDA Wikipedia ③	53.7	68.4	37.4	40.5	59.7	69.5	57.5	70.1	52.3	56.8
LDA Dataset ④	46.2	58.9	36.9	38.7	52.7	55.2	21.5	27.3	51.0	52.3
Character n-grams	48.8	57.0	49.0	51.0	58.5	63.6	60.2	65.6	56.2	58.4
Word Embeddings ⑤	64.6	**88.4**	57.5	**62.5**	63.2	68.3	55.8	59.6	59.5	59.7
Character Embeddings ⑥	**66.5**	87.2	55.5	59.2	63.7	66.4	56.2	61.7	62.4	64.0
①+②	62.1	86.8	54.5	59.5	66.3	75.4	62.5	71.1	59.1	62.7
①+③	63.0	84.8	52.7	57.6	66.1	74.8	63.4	**72.4**	56.8	60.7
①+④	63.3	85.1	53.1	58.3	65.6	74.5	63.1	72.2	57.4	61.0
①+⑤	66.5	84.1	54.0	58.9	67.6	76.5	62.8	72.0	57.0	60.7
①+⑥	65.8	83.0	53.1	58.3	**67.8**	**76.8**	63.3	71.2	**64.3**	**65.6**
①+②+⑤	62.5	85.1	55.8	60.9	66.6	75.6	63.8	72.3	58.2	62.0
①+②+⑥	65.1	86.3	54.5	60.1	66.5	75.9	63.3	72.0	64.3	65.6
①+②+⑤+⑥	64.4	86.1	55.0	59.3	64.8	74.2	**63.8**	72.2	64.3	65.6

Tabelle 89: Document-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	69.0	84.0	34.8	56.6	59.9	62.9	47.4	47.9	42.1	58.2
Grammatical ②	72.0	85.9	37.6	68.9	25.5	17.5	24.3	15.6	49.6	63.0
LDA Wikipedia ③	67.7	84.6	42.6	72.4	58.7	**65.2**	46.5	58.4	43.0	62.8
LDA Dataset ④	67.7	84.6	42.6	72.4	52.9	60.4	50.1	58.3	43.0	62.8
Character n-grams	67.7	84.6	42.6	72.4	52.9	60.4	50.1	58.3	44.2	63.4
Word Embeddings ⑤	67.5	84.5	**43.7**	**73.9**	57.9	62.4	52.3	60.9	47.6	65.0
Character Embeddings ⑥	69.7	85.2	29.7	71.6	53.8	62.2	**54.2**	**62.5**	42.7	63.8
①+②	45.8	77.2	29.7	71.6	54.7	59.6	24.3	15.6	**53.5**	**66.4**
①+③	69.7	85.4	34.8	56.6	50.0	59.2	47.4	47.9	45.8	64.4
①+④	69.7	85.4	34.8	56.6	57.8	64.0	47.8	48.3	44.6	63.8
①+⑤	64.5	83.7	37.1	59.7	**60.8**	64.1	49.8	51.0	44.1	64.3
①+⑥	66.2	84.0	29.7	71.6	47.6	57.9	53.2	57.4	42.7	63.8
①+②+⑤	71.0	85.1	36.3	67.4	56.8	61.2	24.3	15.6	53.0	65.5
①+②+⑥	72.0	85.8	38.6	73.3	25.5	17.5	52.3	61.6	51.4	65.0
①+②+⑤+⑥	**72.8**	**86.3**	35.5	68.4	55.3	60.5	52.6	62.2	51.5	65.5

Tabelle 90: Document-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **CRF**.

Subtask	A		B		C		D		E	
*F*1-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	60.4	78.9	48.6	62.4	59.1	58.9	43.8	44.5	51.8	52.2
Grammatical ②	56.4	83.3	47.9	59.9	63.0	65.1	**55.8**	**64.5**	**57.9**	**58.3**
LDA Wikipedia ③	57.1	83.3	47.1	61.8	61.5	64.0	52.7	61.6	52.0	52.5
LDA Dataset ④	59.4	84.0	47.1	61.8	61.5	64.0	52.7	61.6	52.0	52.5
Character n-grams	57.1	83.3	47.1	61.8	59.6	62.2	52.7	61.6	51.3	51.7
Word Embeddings ⑤	55.9	83.1	**49.8**	**63.0**	61.5	64.4	54.8	63.9	56.9	57.3
Character Embeddings ⑥	59.0	83.5	26.3	51.6	**68.0**	**70.5**	54.0	63.1	55.0	55.6
①+②	61.1	84.2	47.7	59.9	64.8	66.9	55.7	64.1	57.8	58.0
①+③	56.4	75.4	46.7	57.6	59.1	58.9	43.8	44.5	51.4	51.8
①+④	61.1	79.9	48.2	60.1	59.1	58.9	49.0	51.3	53.4	53.7
①+⑤	61.9	**84.8**	47.7	62.6	59.9	59.9	45.5	46.2	53.2	53.6
①+⑥	**63.3**	84.3	26.3	51.6	63.9	65.3	46.4	47.8	57.0	57.5
①+②+⑤	60.6	84.2	46.5	58.1	65.0	67.0	22.6	13.2	56.5	56.8
①+②+⑥	48.8	81.4	43.1	58.1	38.5	48.1	54.5	64.1	56.7	56.9
①+②+⑤+⑥	49.1	81.4	49.7	62.8	60.0	62.5	54.1	64.1	56.7	56.9

Tabelle 91: Document-Tagging Ergebnisse für Datensatz **Both** mit Klassifikator **CRF**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	47.9	83.0	46.2	53.5	57.3	69.5	52.5	63.7	44.6	50.6
Grammatical ②	50.4	83.9	48.2	54.4	53.3	63.6	53.5	63.0	38.6	48.5
LDA Wikipedia ③	**53.5**	**84.3**	47.3	54.5	55.9	68.9	51.8	62.3	48.2	53.9
LDA Dataset ④	53.5	84.3	48.9	54.9	55.9	68.9	51.8	62.3	43.9	51.6
Character n-grams	53.5	84.3	47.3	54.5	59.6	68.4	51.8	62.3	48.2	53.9
Word Embeddings ⑤	51.3	84.3	48.2	**55.3**	59.0	69.5	51.8	63.1	46.4	52.3
Character Embeddings ⑥	47.0	83.5	23.8	39.7	59.3	**70.6**	49.1	64.2	42.3	50.8
①+②	47.7	83.5	**49.4**	54.9	55.1	64.1	52.2	62.2	**50.3**	**55.5**
①+③	47.2	82.8	46.2	53.5	**60.4**	70.5	52.5	63.7	48.6	52.9
①+④	47.0	83.5	46.2	53.5	57.3	69.5	53.6	**65.8**	44.6	50.6
①+⑤	49.9	84.0	48.0	55.1	59.7	70.2	52.9	63.9	43.9	49.6
①+⑥	47.0	83.5	34.3	45.8	56.5	68.7	53.8	65.0	42.4	51.0
①+②+⑤	47.4	83.1	47.7	53.2	53.2	62.8	42.1	61.2	38.6	48.5
①+②+⑥	46.9	83.3	46.1	53.0	59.0	69.1	52.4	62.3	38.6	48.5
①+②+⑤+⑥	48.1	83.2	45.2	52.2	59.2	67.5	**54.3**	63.9	49.6	54.8

Tabelle 92: Document-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **CRF**.

A.3.2 Klassisches Machine Learning: Cross-Platform

In diesem Kapitel werden die einzelnen Cross-Platform Ergebnisse der klassischen Machine Learning Verfahren aufgeführt. Es folgen Tabellen für jeden Klassifikator und Datensatz sowie zusammenfassende Abbildungen. In allen Tabellen sowie Abbildungen wird jeweils der Datensatz des verwendeten Trainingssets angegeben. Der Datensatz des verwendeten Testsets ist für Cross-Platform Untersuchungen immer der jeweils andere Datensatz. Der kombinierte Datensatz Both wurde hierbei nicht untersucht.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	53.3	62.2	38.8	54.6	54.6	70.6	49.9	72.6	28.9	37.0
Grammatical ②	59.3	72.7	**48.6**	56.1	**61.8**	64.7	53.2	68.5	48.1	48.7
LDA Wikipedia ③	59.6	73.9	36.5	48.1	56.3	58.3	53.6	69.5	41.7	43.5
LDA Dataset ④	58.4	74.3	35.8	49.6	48.3	49.3	47.2	67.9	44.8	45.5
Character n-grams	55.9	68.5	41.6	53.4	57.1	59.7	48.9	71.6	33.9	39.9
Word Embeddings ⑤	61.6	78.4	41.2	58.9	51.4	55.2	**59.1**	**73.9**	47.1	48.1
Character Embeddings ⑥	63.1	**86.1**	37.7	55.8	48.6	50.1	57.7	72.7	46.5	47.7
①+②	55.8	67.1	37.8	53.5	54.4	**71.7**	49.1	72.8	30.2	38.6
①+③	58.7	72.2	32.2	47.5	55.3	57.1	55.1	67.7	45.5	46.4
①+④	58.3	70.9	40.3	55.0	56.4	66.4	52.5	73.2	**52.8**	**53.2**
①+⑤	61.7	78.6	39.1	58.2	54.5	57.4	56.3	73.1	50.8	51.0
①+⑥	62.2	78.1	40.1	55.6	58.5	71.4	56.8	67.0	41.6	42.9
①+②+⑤	58.9	74.8	42.5	**59.5**	60.5	71.7	58.1	73.3	50.9	51.3
①+②+⑥	**65.7**	83.6	40.3	59.5	59.6	68.3	58.3	71.9	51.3	51.9
①+②+⑤+⑥	63.8	81.8	41.7	57.6	60.0	67.2	56.9	72.3	49.0	49.0

Tabelle 93: Cross-Platform Document-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	60.6	86.6	34.2	73.6	58.0	66.7	58.7	64.0	45.0	45.8
Grammatical ②	62.0	**87.1**	39.1	72.6	51.1	56.7	57.9	66.3	45.8	45.8
LDA Wikipedia ③	60.6	84.1	38.8	67.6	46.9	47.5	57.2	59.3	39.4	42.3
LDA Dataset ④	**69.7**	82.9	34.0	68.4	44.3	44.7	60.6	66.2	48.6	49.8
Character n-grams	48.8	84.6	39.7	76.9	48.9	48.9	53.8	63.0	39.5	39.5
Word Embeddings ⑤	51.2	85.2	44.4	72.7	**65.0**	67.4	61.1	70.1	48.4	49.0
Character Embeddings ⑥	54.5	85.6	44.8	66.6	58.4	62.4	64.3	71.1	47.8	49.8
①+②	55.9	86.0	37.9	76.9	51.0	58.2	61.2	65.9	46.6	49.0
①+③	66.1	84.6	39.2	76.2	62.1	**68.1**	56.8	61.3	49.6	53.4
①+④	63.6	86.0	35.5	76.3	57.0	57.4	59.1	62.6	50.9	52.6
①+⑤	52.6	85.5	43.9	74.9	59.8	68.1	61.1	71.1	46.4	47.8
①+⑥	54.6	85.7	42.9	68.4	57.6	65.2	63.3	69.8	**52.2**	**54.5**
①+②+⑤	51.2	85.2	44.8	**77.1**	61.9	65.2	60.9	70.5	48.2	49.0
①+②+⑥	53.9	85.6	**44.9**	70.1	58.6	63.8	63.3	70.0	43.6	44.3
①+②+⑤+⑥	55.0	85.5	44.5	74.3	61.0	68.1	**64.9**	**71.4**	50.9	54.2

Tabelle 94: Cross-Platform Document-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Random Forest**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	48.7	88.9	39.0	**59.5**	59.3	65.8	56.5	70.9	46.5	47.7
Grammatical ②	58.1	73.3	**51.5**	54.5	43.4	43.4	53.9	62.7	27.0	37.0
LDA Wikipedia ③	48.2	61.8	11.3	18.0	56.2	63.3	21.5	27.3	34.9	40.6
LDA Dataset ④	53.5	75.4	34.2	51.5	31.4	31.9	51.0	61.0	51.2	51.3
Character n-grams	57.5	81.7	47.7	55.0	50.3	50.4	51.0	55.6	46.9	47.7
Word Embeddings ⑤	57.0	84.4	51.3	57.5	48.8	49.0	**59.4**	62.8	**58.3**	**59.7**
Character Embeddings ⑥	**61.5**	75.3	50.9	58.6	51.9	52.7	57.8	62.4	52.5	52.6
①+②	58.7	81.0	37.9	59.0	59.5	63.6	55.7	70.4	40.2	43.5
①+③	59.0	81.7	40.9	58.7	56.4	58.5	56.5	71.0	48.6	49.0
①+④	59.2	82.1	40.0	53.5	59.5	64.4	56.5	70.9	56.1	56.2
①+⑤	48.7	**89.0**	43.5	55.1	58.9	63.6	56.0	**72.4**	42.8	44.8
①+⑥	60.2	77.7	41.6	53.9	62.0	70.3	54.9	62.8	49.2	49.7
①+②+⑤	57.9	78.9	43.2	55.3	61.1	67.5	55.3	71.9	30.0	38.3
①+②+⑥	58.7	74.2	41.9	54.5	**62.5**	**70.6**	55.1	63.0	39.7	42.9
①+②+⑤+⑥	60.5	78.3	42.9	56.6	61.5	67.8	55.5	71.7	44.2	46.4

Tabelle 95: Cross-Platform Document-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	54.4	85.5	29.7	**80.3**	25.4	34.0	58.6	64.2	20.4	25.7
Grammatical ②	**71.6**	**87.3**	42.0	64.0	50.5	61.7	56.9	57.6	48.9	53.4
LDA Wikipedia ③	58.3	80.8	36.1	65.9	52.1	52.5	53.5	56.0	37.9	38.7
LDA Dataset ④	54.7	74.2	36.5	60.2	49.5	52.5	54.3	59.1	52.1	63.2
Character n-grams	48.9	84.9	37.0	79.0	25.4	34.0	49.7	63.1	39.5	39.5
Word Embeddings ⑤	71.5	83.6	**44.7**	70.5	55.6	57.4	62.3	68.5	**59.4**	**71.1**
Character Embeddings ⑥	70.9	85.0	40.7	58.8	**64.4**	**70.2**	**64.5**	**69.1**	56.1	62.5
①+②	59.0	86.4	37.0	79.8	33.0	36.9	60.0	65.8	45.8	46.2
①+③	48.9	84.9	38.5	76.2	25.4	34.0	58.6	64.3	20.4	25.7
①+④	54.3	85.3	37.1	79.8	34.0	37.6	60.6	65.9	31.5	33.2
①+⑤	48.9	84.9	37.2	80.0	50.3	50.4	60.6	66.1	31.5	33.2
①+⑥	69.1	84.5	41.2	70.4	54.8	62.4	62.6	67.5	52.1	56.5
①+②+⑤	61.5	86.2	37.1	79.8	47.5	47.5	63.2	68.3	45.6	46.2
①+②+⑥	69.2	84.6	40.6	70.7	54.8	62.4	62.4	67.9	52.1	56.5
①+②+⑤+⑥	69.2	84.6	40.6	70.7	54.8	62.4	62.3	67.8	52.1	56.5

Tabelle 96: Cross-Platform Document-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	56.3	67.9	41.9	52.9	59.4	67.8	54.9	71.0	55.2	55.2
Grammatical ②	59.0	74.0	**50.9**	53.5	41.7	41.7	42.1	**72.7**	27.0	37.0
LDA Wikipedia ③	26.7	26.9	10.2	17.6	53.8	58.8	21.5	27.3	59.5	60.4
LDA Dataset ④	44.2	57.2	32.2	34.8	50.3	52.4	47.7	54.4	53.5	54.2
Character n-grams	37.3	40.3	35.2	36.8	53.4	56.9	54.1	70.6	51.8	51.9
Word Embeddings ⑤	55.7	**85.1**	44.7	49.9	45.0	45.1	21.5	27.3	**60.0**	**62.3**
Character Embeddings ⑥	**61.9**	79.1	43.5	**57.4**	49.8	51.5	54.0	59.5	51.8	51.9
①+②	58.9	71.3	43.5	54.2	59.5	67.5	53.4	69.8	44.8	46.8
①+③	56.6	68.3	42.7	52.8	**59.9**	68.1	54.4	70.8	55.5	55.5
①+④	56.4	68.1	42.8	52.6	59.7	67.8	54.1	70.6	56.2	56.2
①+⑤	57.3	69.5	41.6	52.8	59.4	69.5	54.9	72.2	45.3	46.4
①+⑥	59.9	77.8	43.0	52.8	59.4	69.5	56.3	63.1	40.7	43.2
①+②+⑤	60.4	73.7	42.0	54.1	59.1	69.2	54.1	70.9	44.8	46.8
①+②+⑥	60.2	77.9	42.4	53.5	51.6	**71.7**	56.5	63.2	48.4	48.7
①+②+⑤+⑥	60.5	74.2	42.8	53.7	59.4	69.2	**56.5**	63.2	48.4	48.7

Tabelle 97: Cross-Platform Document-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	61.6	79.7	39.0	67.9	60.4	62.4	64.3	68.4	48.2	49.0
Grammatical ②	13.8	15.9	41.8	64.7	48.4	57.4	56.1	58.5	54.4	67.2
LDA Wikipedia ③	54.4	66.2	32.6	49.8	45.1	46.1	59.8	67.8	42.5	43.1
LDA Dataset ④	39.9	46.6	29.7	**80.3**	47.6	49.6	24.3	32.1	48.5	60.1
Character n-grams	61.7	70.1	35.1	65.9	60.4	68.8	58.3	59.6	53.3	56.1
Word Embeddings ⑤	65.1	86.7	**43.3**	70.2	61.3	63.8	57.3	58.9	**61.6**	**71.1**
Character Embeddings ⑥	66.9	**87.1**	38.9	62.0	**61.4**	**70.9**	58.2	63.1	56.7	64.0
①+②	55.7	74.0	40.1	70.2	55.0	57.4	64.7	69.1	46.4	46.6
①+③	60.9	79.0	39.2	68.4	58.0	60.3	64.3	68.4	46.1	47.0
①+④	61.2	78.8	39.8	68.8	59.1	61.0	64.6	68.9	48.4	49.4
①+⑤	66.7	82.1	39.4	68.7	58.6	61.0	64.1	68.3	48.5	49.4
①+⑥	**67.1**	83.0	42.5	67.9	58.6	61.0	64.7	68.5	53.0	56.5
①+②+⑤	63.8	81.1	40.0	70.0	53.7	55.3	64.9	69.7	45.7	46.2
①+②+⑥	62.3	80.8	42.9	68.4	53.4	54.6	64.8	69.5	53.6	56.9
①+②+⑤+⑥	56.0	74.7	42.9	68.4	54.6	56.0	**65.2**	**69.8**	53.6	56.9

Tabelle 98: Cross-Platform Document-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **Support Vector Machine: Linearer Kernel**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	55.2	78.9	38.6	43.7	52.8	59.9	39.9	41.5	40.2	37.6
Grammatical ②	57.9	79.4	**44.1**	50.9	42.4	62.4	21.5	11.7	45.0	43.5
LDA Wikipedia ③	58.5	82.4	38.3	47.0	54.2	57.1	45.5	62.0	37.7	33.6
LDA Dataset ④	58.5	82.4	38.3	47.0	53.3	56.4	51.6	61.2	37.7	33.6
Character n-grams	58.5	82.4	38.3	47.0	53.3	56.4	51.6	61.2	41.2	37.8
Word Embeddings ⑤	57.8	83.2	37.6	47.1	50.2	54.6	**51.7**	63.4	34.3	29.3
Character Embeddings ⑥	58.0	81.3	23.8	39.7	56.5	60.2	49.1	63.4	29.4	22.9
①+②	47.0	83.5	23.8	39.7	**57.9**	**63.7**	21.5	11.7	44.4	42.2
①+③	58.6	82.1	38.6	43.7	51.7	54.1	39.9	41.5	41.9	38.6
①+④	58.6	82.1	38.6	43.7	53.1	56.5	39.8	41.4	40.4	36.8
①+⑤	55.2	**83.8**	38.8	44.5	51.2	56.6	44.4	47.6	31.4	25.3
①+⑥	58.5	82.8	23.8	39.7	40.2	38.3	51.0	59.1	27.1	20.1
①+②+⑤	57.8	78.5	42.2	49.0	57.2	62.9	21.5	11.7	**47.3**	**46.0**
①+②+⑥	**59.6**	79.9	39.6	48.5	42.4	62.4	50.2	**64.7**	45.0	43.2
①+②+⑤+⑥	59.0	79.4	43.6	**51.1**	57.4	63.2	48.9	63.9	46.7	44.9

Tabelle 99: Cross-Platform Document-Tagging Ergebnisse für Datensatz **BRK** mit Klassifikator **CRF**.

Subtask	A		B		C		D		E	
$F1$-Score	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro	Macro	Micro
Unigram ①	49.4	77.8	33.8	70.1	**54.0**	**56.1**	50.5	56.2	36.9	36.5
Grammatical ②	50.1	78.6	**39.3**	67.9	49.0	53.0	56.5	62.3	20.3	10.3
LDA Wikipedia ③	**52.9**	**79.3**	35.2	70.6	44.8	43.7	49.6	55.6	45.1	46.4
LDA Dataset ④	52.9	79.3	33.4	67.5	44.8	43.7	49.6	55.6	32.6	30.1
Character n-grams	52.9	79.3	35.2	70.6	50.0	53.3	49.6	55.6	45.1	46.4
Word Embeddings ⑤	50.3	78.8	37.1	**72.4**	50.9	52.5	49.2	56.5	36.9	37.3
Character Embeddings ⑥	45.8	77.2	29.7	71.6	50.7	50.1	48.8	59.6	30.5	25.9
①+②	45.6	77.0	35.7	64.5	49.7	54.1	**57.6**	**62.9**	**46.7**	**50.8**
①+③	49.8	77.8	33.8	70.1	48.5	52.8	50.5	56.2	42.4	43.2
①+④	45.8	77.2	33.8	70.1	54.0	56.1	49.0	57.8	36.9	36.5
①+⑤	49.5	78.5	36.6	72.2	46.7	50.4	50.5	56.6	36.9	36.0
①+⑥	45.8	77.2	34.0	72.4	47.0	47.8	52.5	59.5	30.5	25.9
①+②+⑤	45.5	76.7	37.0	64.5	49.7	54.1	40.4	54.9	20.3	10.3
①+②+⑥	45.6	77.0	36.4	65.9	49.7	53.2	54.6	60.5	20.3	10.3
①+②+⑤+⑥	46.0	76.7	36.6	65.7	50.1	54.2	55.7	61.6	45.5	49.7

Tabelle 100: Cross-Platform Document-Tagging Ergebnisse für Datensatz **THF** mit Klassifikator **CRF**.

References

Sami Abu-El-Haija (2018). *Logistic Regression*. Accessed: 2018-09-10. URL: %5Curl %7Bhttp://www.haija.org/derivation_logistic_regression.pdf%7D.

David M Blei, Andrew Y Ng und Michael I Jordan (2002). „Latent Dirichlet Allocation“. In: *Advances in neural information processing systems*, S. 601–608.

David M Blei, Andrew Y Ng und Michael I Jordan (2003). „Latent Dirichlet Allocation“. In: *Journal of Machine Learning Research* 3.Jan, S. 993–1022.

Piotr Bojanowski, Edouard Grave, Armand Joulin und Tomas Mikolov (2017). „Enriching Word Vectors with Subword Information“. In: *Transactions of the Association for Computational Linguistics* 5, S. 135–146.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall und W Philip Kegelmeyer (2002). „SMOTE: Synthetic Minority Over-sampling Technique“. In: *Journal of Artificial Intelligence Research* 16, S. 321–357.

Edwin Chen (2018a). *Chen CRF*. Accessed: 2018-09-10. URL: %5Curl%7Bhttp://blog.echen.me/2012/01/03/introduction-to-conditional-random-fields/%7D.

Edwin Chen (2018b). *Introduction to Latent Dirichlet Allocation*. Accessed: 2018-09-17. URL: %5Curl%7Bhttp://blog.echen.me/2011/08/22/introduction-to-latent-dirichlet-allocation/%7D.

Stefan Conrad (2016). *Knowledge Discovery in Databases (Heinrich-Heine-Universität Düsseldorf, WS 2016/2017)*. Accessed: 2018-12-26. URL: %5Curl%7Bhttps://dbs.cs.uni-duesseldorf.de/lehre/veranst.php?veranst=84%7D.

Corinna Cortes und Vladimir Vapnik (1995). „Support-Vector Networks“. In: *Machine learning* 20.3, S. 273–297.

Katharina Esau (2018). „Capturing Citizens' Values: On the Role of Narratives and Emotions in Digital Participation“. In: *Analyse & Kritik* 40.1, S. 55–72.

Vanessa Wei Feng und Graeme Hirst (2011). „Classifying Arguments by Scheme“. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, S. 987–996.

James B Freeman (2011). *Dialectics and the Macrostructure of Arguments: A Theory of Argument Structure*. Bd. 10. Walter de Gruyter.

Yoav Goldberg (2016). „A Primer on Neural Network Models for Natural Language Processing“. In: *Journal of Artificial Intelligence Research* 57, S. 345–420.

Yoav Goldberg und Omer Levy (2014). „word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method“. In: *arXiv preprint arXiv:1402.3722*.

Theodosis Goudas, Christos Louizos, Georgios Petasis und Vangelis Karkaletsis (2014). „Argument Extraction from News, Blogs, and Social Media“. In: *Hellenic Conference on Artificial Intelligence*. Springer, S. 287–299.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin und Tomas Mikolov (2018). „Learning word vectors for 157 languages“. In: *arXiv preprint arXiv:1802.06893*.

Chinnappa Guggilla, Tristan Miller und Iryna Gurevych (2016). „CNN- and LSTM-based Claim Classification in Online User Comments“. In: *COLING*.

Jiang Guo (2013). „BackPropagation Through Time“. In: *Technical Report*.

Ivan Habernal und Iryna Gurevych (2016). „Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM“. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Bd. 1, S. 1589–1599.

Ivan Habernal und Iryna Gurevych (2017). „Argumentation Mining in User-Generated Web Discourse“. In: *Computational Linguistics* 43.1, S. 125–179.

Haibo He, Yang Bai, Edwardo A Garcia und Shutao Li (2008). „ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning“. In: *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, S. 1322–1328.

David Hitchcock und Bart Verheij (2006). *Arguing on the model*. Springer.

Tin Kam Ho (1995). „Random decision forests“. In: *Document analysis and recognition, 1995., proceedings of the third international conference on*. Bd. 1. IEEE, S. 278–282.

Sepp Hochreiter und Jürgen Schmidhuber (1997). „Long Short-Term Memory“. In: *Neural computation* 9.8, S. 1735–1780.

Armand Joulin, Edouard Grave, Piotr Bojanowski und Tomas Mikolov (Apr. 2017). „Bag of Tricks for Efficient Text Classification“. In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, S. 427–431.

Klaus Krippendorff (2004). *Content analysis: An introduction to its methodology*.

Klaus Krippendorff (2011). *Computing Krippendorff's Alpha-Reliability*.

John D. Lafferty, Andrew McCallum und Fernando C. N. Pereira (2001). „Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data“. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. Morgan Kaufmann Publishers Inc., S. 282–289.

Guillaume Lemaître, Fernando Nogueira und Christos K. Aridas (2017). „Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning“. In: *Journal of Machine Learning Research* 18.17, S. 1–5.

Matthias Liebeck (2018). „Automated Discussion Analysis in Online Participation Projects“. Diss.

Matthias Liebeck, Katharina Esau und Stefan Conrad (2016). „What to Do with an Airport? Mining Arguments in the German Online Participation Project Tempelhofer Feld“. In: *Proceedings of the Third Workshop on Argument Mining (ArgMining2016)*. Association for Computational Linguistics, S. 144–153.

Matthias Liebeck, Katharina Esau und Stefan Conrad (2017). „Text Mining für Online-Partizipationsverfahren: Die Notwendigkeit einer maschinell unterstützten Auswertung“. In: *HMD Praxis der Wirtschaftsinformatik: Vol. 54, No. 4*, S. 544–562.

Chris McCormick (Apr. 2016). *Word2Vec Tutorial - The Skip-Gram Model*. Accessed: 2018-09-13. URL: %5Curl%7Bhttp://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/%7D.

Chris McCormick (Jan. 2017). *Word2Vec Tutorial Part 2 - Negative Sampling*. Accessed: 2018-09-13. URL: %5Curl%7Bhttp://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/%7D.

Tomas Mikolov, Kai Chen, Greg Corrado und Jeffrey Dean (2013a). „Efficient Estimation of Word Representations in Vector Space“. In: *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado und Jeff Dean (2013b). „Distributed Representations of Words and Phrases and their Compositionality“. In: *Advances in neural information processing systems*, S. 3111–3119.

Jonathan Milgram, Mohamed Cheriet und Robert Sabourin (2006). „“One Against One” or “One Against All”: Which One is Better for Handwriting Recognition with SVMs?“ In: *Tenth International Workshop on Frontiers in Handwriting Recognition*. SuviSoft.

Raquel Mochales und Marie-Francine Moens (2011). „Argumentation Mining“. In: *Artificial Intelligence and Law* 19.1, S. 1–22.

Hien M Nguyen, Eric W Cooper und Katsuari Kamei (2011). „Borderline Over-sampling for Imbalanced Data Classification“. In: *International Journal of Knowledge Engineering and Soft Data Paradigms* 3.1, S. 4–21.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot und E. Duchesnay (2011). „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12, S. 2825–2830.

J. Ross Quinlan (1986). „Induction of Decision Trees“. In: *Machine learning* 1.1, S. 81–106.

Lance A Ramshaw und Mitchell P Marcus (1999). „Text Chunking using Transformation-Based Learning“. In: *Natural language processing using very large corpora*. Springer, S. 157–176.

Lev Ratinov und Dan Roth (2009). „Design challenges and misconceptions in named entity recognition“. In: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*. Association for Computational Linguistics, S. 147–155.

Jodi Schneider und Adam Z Wyner (2012). „Identifying Consumers' Arguments in Text.“ In: *SWAIE*, S. 31–42.

Mike Schuster und Kuldip K Paliwal (1997). „Bidirectional Recurrent Neural Networks“. In: *IEEE Transactions on Signal Processing* 45.11, S. 2673–2681.

sklearn Random Forest (o.D.). Accessed: 2018-12-25. URL: %5Curl % 7Bhttps : / / scikit - learn . org / stable / modules / generated / sklearn . ensemble . RandomForestClassifier.html%7D.

Christian Stab und Iryna Gurevych (2014). „Annotating Argument Components and Relations in Persuasive Essays“. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, S. 1501–1510.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou und Jun'ichi Tsujii (2012). „BRAT: a Web-based Tool for NLP-Assisted Text Annotation“. In: *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, S. 102–107.

Charles Sutton, Andrew McCallum et al. (2012). „An Introduction to Conditional Random Fields“. In: *Foundations and Trends® in Machine Learning* 4.4, S. 267–373.

Toulmin Model of Argumentation (2018). Accessed: 2018-08-06. URL: %5Curl % 7Bhttp : //individual.utoronto.ca/ecolak/EBM/evidence_and_eikos/models_of_argumentation/toulmin/toulmin.htm%7D.

Stephen Toulmin (2003). „The Uses of Argument“. In: *Cambridge: Cambridge UP*.

Andrew Viterbi (1967). „Error bounds for convolutional codes and an asymptotically optimum decoding algorithm“. In: *IEEE transactions on Information Theory* 13.2, S. 260–269.

Henning Wachsmuth, Martin Trenkmann, Benno Stein, Gregor Engels und Tsvetomira Palakarska (2014). „A Review Corpus for Argumentation Analysis“. In: *International Conference on Intelligent Text Processing and Computational Linguistics*. Springer, S. 115–127.

Jaya Kumar Yogan, Ong Sing Goh, Basiron Halizah, Hea Choon Ngo und C Puspalata (2016). „A Review on Automatic Text Summarization Approaches“. In: *Journal of Computer Science* 12.4, S. 178–190.

Abbildungsverzeichnis

Tabellenverzeichnis