Lehrstuhl Angewandte Informatik IV
Datenbanken und Informationssysteme
Prof. Dr.-Ing. Stefan Jablonski

Institut für Angewandte Informatik
Fakultät für Mathematik, Physik und Informatik
Universität Bayreuth

Bachelorarbeit

Philipp Scholz

23.April 2019
Version: Final

Universität Bayreuth

Fakultät Mathematik, Physik, Informatik
Institut für Informatik
Lehrstuhl für Angewandte Informatik IV

NLP-Plattform: Integration und Evaluation von POS-Tagging-Algorithmen

Bachelorarbeit

Philipp Scholz

1. Reviewer Dr. Lars Ackermann
Fakultät Mathematik, Physik, Informatik
Universität Bayreuth

2. Reviewer Prof. Dr.-Ing. Stefan Jablonski
Fakultät Mathematik, Physik, Informatik
Universität Bayreuth

Supervisors Lars Ackermann und Stefan Jablonski

23.April 2019

Philipp Scholz
Bachelorarbeit
NLP-Plattform: Integration und Evaluation von POS-Tagging-Algorithmen, 23.April 2019
Reviewers: Dr. Lars Ackermann and Prof. Dr.-Ing. Stefan Jablonski
Supervisors: Lars Ackermann and Stefan Jablonski

Universität Bayreuth
Lehrstuhl für Angewandte Informatik IV
Institut für Informatik
Fakultät Mathematik, Physik, Informatik
Universitätsstrasse 30
95447 Bayreuth
Germany

Abstrakt/Abstract

Deutsch

Im Themenbereich *Natural Language Processing* (kurz NLP) versucht die Informatik, natürliche Sprachen für Algorithmen zugänglich und interpretierbar zu machen. Ein wichtiger Teil von NLP ist die Identifizierung der syntaktischen Bedeutung von Wörtern (*Parts of Speech*, kurz POS) in gesprochener Sprache, und deren Zuweisung in Form von Tags (POS-Tagging). Für diese Aufgabe existiert eine Vielzahl unterschiedlich leistungsfähiger und robuster Algorithmen.
Im Rahmen dieser Arbeit wurde eine Sammlung solcher POS-Tagging-Algorithmen zusammen mit einem Evaluationssystem als Operatoren im Programm RapidMiner (zur Verfügung gestellt von RapidMiner GmbH) implementiert.

English

The field of Computer Science called *Natural Language Processing* (NLP) attempts to make natural language accessible for machines and algorithms. One of the major parts of NLP is the identification of the syntactic roles of words and symbols (*Parts of Speech, POS* in short) in spoken language, and tagging them as such POS (POS-Tagging). There is a multitude of differently capable and robust algorithms for this task.
In this project, such POS-Tagging algorithms and a system to evaluate them is implemented in form of operators in an extension to the program RapidMiner (by RapidMiner GmbH).

Inhaltsverzeichnis

1 Einleitung

:TODO :NC

Keine bekannte Lebensform hat eine mit der menschlichen Sprache vergleichbar komplexe Form von Informationsaustausch entwickelt [C R18]. Von selbst ergibt sich die Frage, ob und wie Sprache maschinell verarbeitet werden kann, um sie unter anderem zu interpretieren oder zusammenzufassen. Antworten auf diese Problemstellung liefert das Teilgebiet der Informatik *Natural Language Processing* (Dt. Verarbeitung natürlicher Sprachen, kurz NLP). NLP gliedert sich in viele Teilbereiche: Strukturelle Analyse, Semantik, Phonetik und einige weitere. In dieser Arbeit konzentrieren wir uns auf einen wichtigen Bestandteil der strukturellen Analyse, dem korrekten Identifizieren von syntaktischen Rollen von Wörtern und Symbolen (*Parts of Speech*, kurz POS), bezeichnet als *POS-Tagging* [Smi11].
Ein Algorithmus, der POS-Tagging betreibt (POS-Tagger), nimmt die Sprache in Textform an und gibt ihn üblicherweise mit Tags versehen wieder aus, wie Beispielsweise der Text

I like the blue house.

zu folgendem verarbeitet wird:

*I****PRONOUN** *like****VERB** *the****DET** *blue****ADJ** *house****NOUN** .**.**

Wie die Tatsächliche Ausgabe eines Taggers exakt formatiert wird und welche Tags auftreten, wird später angesprochen. Wichtiger ist hingegen, dass POS-Tagger diese Tags nicht garantiert korrekt wählen (siehe Abschnitt 1.2). Es ist also von Interesse, deren Performance zu bewerten.

1.1 Problemstellung dieser Arbeit

Die Datenverarbeitungs-Plattform *RapidMiner* :NC bietet im Rahmen der Erweiterung *Text Processing* :NC Funktionen (*Operatoren*), um Texte einzulesen und zu verarbeiten. NLP ist in RapidMiners Textverarbeitungs-Erweiterungen jedoch weitgehend noch nicht implementiert. Ziel dieser Arbeit ist es, in Form einer auf *Text Processing*

aufbauenden Erweiterung sowohl POS-Tagger zu implementieren, als auch ein Evaluationsrahmenwerk, mit dem deren Performance gemessen werden kann.

:TODO ?

Die implementierte Erweiterung liefert :TODO[anzahl] Tagging-Operatoren, :TODO[anzahl] Evaluationsoperatoren, ein spezialisiertes Übergabeformat für Tagging-Ergebnisse und eine Standardisierung für verwendete Tagsets :TODO[genauer?]. Gleichzeitig sind die Operatoren in ihrem Ein- und Ausgangsformat weitgehend kompatibel mit der Erweiterung *Text Processing*.

1.2 Part-of-Speech-Tagging allgemein

Betrachtet man das Wort „like“ aus dem einleitenden Beispiel, dann fällt auf, dass es alternativ zum Verb „mögen“ auch als Präposition „wie“ interpretiert werden könnte, auch wenn intuitiv schnell erkennbar wird, dass letztere Variante falsch ist. Diese Uneindeutigkeit (*Ambiguität*) ist das zentrale zu lösende Problem für POS-Tagger [Smi11]; Im Gegensatz zum Menschen kann ein Algorithmus Ambiguitäten nicht intuitiv auflösen. Aus diesem Grund arbeiten POS-Tagger nicht zwingend vollständig korrekt.
Während NLP viele andere Analyseaufgaben neben POS-Tagging zusammenfasst, sind diese bei moderneren und komplexeren NLP-Algorithmen nicht mehr strikt von POS-Tagging trennbar, wenn die Performance des Taggers maximiert werden soll :NC. Zusatzinformationen, die parallel erarbeitet werden können, wie z.B. die Satzstruktur (u.A. Identifizierung von Teilsätzen), können das auflösen von Ambiguitäten erheblich erleichtern. Zur Vereinfachung betrachten wir aber in dieser Arbeit nur POS-Tagging selbst, welches typischerweise in folgende zwei Arbeitsschritte unterteilt wird [Smi11]:

Sequenzierung: Zuerst muss bestimmt werden, welche Teile des angegebenen Dokuments jeweils ein Tag erhalten. Hierzu wird jedes Wort und jedes zusammenhängende Satzzeichen als ein *Token* ausgegeben. Die Reihenfolge der Wörter bleibt als Reihenfolge der Token hierbei erhalten.

Tagging: Mit Hilfe von statistischen, linguistischen und rechnerischen Methoden wird jedem Token ein POS-Tag zugewiesen.

Es folgen noch ein paar weitere Erläuterungen und Definitionen:

1.2.1 Tagset

Um einheitliche Verarbeitung und Vergleichbarkeit zu ermöglichen, werden die Tags in *Tagsets* definiert, wie zum Beispiel dem des Penn-Treebank-Projekts [Sta03] [MP 93].

1.2.2 Korpus und Treebank

Als *Korpus* bezeichnet man eine simple Ansammlung von Text. Der *WSJ-Korpus* :NC beispielsweise ist eine Sammlung von Artikeln der Nachrichtenartikel des *Wall Street Journal*.
Korpora, die zu Token-Ketten sequenzialisiert wurden und deren Token mit (POS-) Tags versehen wurden, werden als annotierte Korpora oder *Treebanks* bezeichnet. Zusätzlich ist eine Treebank mit anderen Informationen wie der Satzstruktur versehen, diese sind hier aber nicht weiter relevant. Treebanks sind in der Regel zu nahezu 100% korrekt annotiert. Ist dies der Fall, spricht man auch von einem *Goldstandard*. Da Goldstandards manuell korrigiert werden, können seltene Fehler auftreten. Diese sind allerdings zu wenige, um als relevant angesehen zu werden. :NC

1.3 Aufbau der Arbeit

Kapitel 2 betrachtet einige Implementierungen von Part-of-Speech-Taggern und deren Evaluation in anderen Projekten.

Kapitel 3 erläutert die Methodik und behandelten Probleme in diesem Projekt.

Kapitel 4 beschreibt detailliert die Implementierung und deren Zielsetzung der RapidMiner-Erweiterung.

Kapitel 5 führt Evaluationen der implementierten POS-Tagger anhand eines gewählten Goldstandards auf.

2 Verwandte Arbeiten

In diesem Kapitel soll auf einige Arbeiten verwiesen werden, in denen Evaluationen von vorgestellten POS-Taggern stattfinden. Es wird besonders beachtet, *wie* und nach welchen Metriken diese Ergebnisse berechnet werden.

2.1 Stanford Tagger

In einem Artikel über den Stanford-Tagger, der zu POS-Tagging fähig ist, wird auch mit Daten von Sektionen aus dem annotierten *Wall Street Journal* Korpus verglichen, um die Qualität der Taggingergebnisse zu prüfen. Hierzu werden unter Anderem die Metriken *Per-Tag-Accuracy* und *Per-Sentence-Accuracy* (siehe Kap. 3.4) genutzt citePaper:StanfordTagger.

2.2 P. Paroubek: Evaluating Part-of-Speech Tagging and Parsing

In dieser Arbeit ([Par07])

3 Konzept

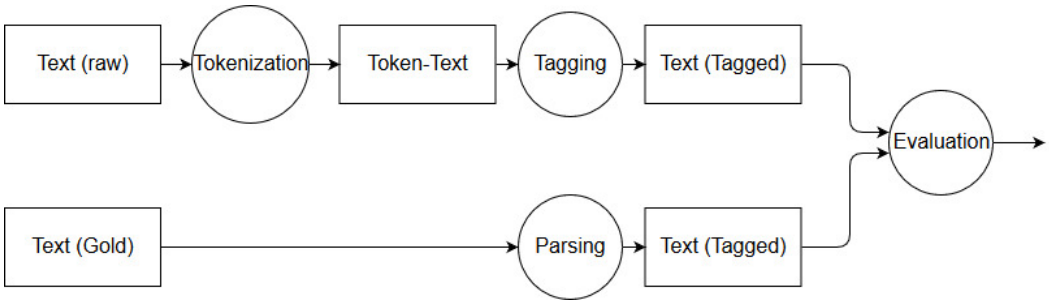

Abb. 3.1: Datenfluss für einen typischen Prozess von Rohdaten bis zur Evaluation

Aus Abbildung 3.1 können die Arbeitsschritte und Daten entnommen werden, die für Tagging und Evaluation modelliert werden müssen. Die folgenden Kapitel behandeln diese Schritte sowie insbesondere deren Eingaben und Ergebnisse.

3.1 Sequenzierung

Bei der Sequenzierung (*Tokenization*) wird zwischen Wort- und Satzsequenzierung unterschieden [Smi11]. Satzsequenzierung beschäftigt sich mit einer Zerlegung des Textes in Sätze, welche für spätere Evaluation (siehe :TODO) nützlich sein kann. Für Part-of-Speech-Tagging ist allerdings vorwiegend die Wort-Sequenzierung wichtig. Außerdem sollte eine Satz-Sequenzierung vor dem Tagging nicht stattfinden, da Tagger ggf. Kontextinformationen aus anderen Sätzen nutzen könnten, was die Auflösung von Ambiguitäten weiter unterstützen könnte.

Da Tagging-Algorithmen sowohl nicht fehlerfrei arbeiten als auch in ihrem Ergebnisformat voneinander - und insbesondere von Goldstandards - abweichen können, entstehen Unsicherheiten beim Vergleich dieser Sequenzen.

3.1.1 Tokenization-Unterschiede

Ein Token ist ein Abschnitt, zu dem ein Tag gehört, typischerweise einzelne Wörter oder Satzzeichen. Formal wird aus dem zusammenhängenden Wort beziehungsweise Text *w* eine Menge *M* von *n* Teilwörtern c_i

$$M = \{c_1, c_2, ..., c_n\}$$

generiert. POS-Tagger wie der von NLP4J :NC und der Stanford University :NC übernehmen diese Zerlegung selbst. Nehmen wir nun an, *M* wäre die Zerlegung von *w* in einem Goldstandard, und ein Tagger produziert ein Ergebnis mit der Zerlegung:

$$M_{tag} = \{d_1, d_2, ..., d_m\}$$

Hierbei sind d_i wieder Teilworte und *m* die Anzahl dieser. Spätestens wenn nun $n \neq m$ gilt, wird klar, dass *M* und M_{tag} nicht mehr einfach iterativ verglichen werden können, da ab der Stelle *p*, wo der Text unterschiedlich geteilt wurde, $c_i \neq d_i$ sein kann, wobei $(p \leq i \leq min\{n, m\})$. Es ist sogar nicht auszuschließen, dass solche Fehler bereits vorher passieren, obwohl weiterhin $n = m$ gilt.
Eines der Ziele bei der Implementierung ist also, den Vergleich von inhomogenen Token-Mengen robust gegen solche Fehler zu machen.

3.1.2 Externe Tokenization

Um alternativ Probleme wie im vorigen Abschnitt zu verhindern, kann alternativ die Sequenzierung extern übernommen werden. D.h. dass den Taggern der Text bereits als Token-Kette übergeben wird. Hierzu muss entweder ein Tokenizer entwickelt werden, der den Taggern Tokens übergibt, die mit ihrem verlangten Format konform sind, oder im Falle eines Goldstandards kann die Token-Zerlegung aus dem annotierten Korpus selbst entnommen werden.
Die meisten (alle implementierten) Tagger unterstützen diese Funktionalität.

3.2 Ergebnisformat

Abhängig vom Tagger können Ergebnisse sehr Informationsreich sein. Der LingPipe-POS-Tagger liefert beispielsweise pro Token (Wort oder Symbol) nicht nur nicht nur das aus seiner Sicht plausibelste Tag (*First-Best*), sondern auch *n-1* weitere, nachstehende Optionen (*N-Best*), wobei *n* gewählt werden kann :NC . Eine solche Menge an Metainformationen kann möglicherweise von bereitgestellten Datenstruk-

turen seitens RapidMiner nur schwer kodiert werden. Eine speziell für Ergebnisse entworfene Datenstruktur bietet neue Freiheiten:

Satz-Sequenzierung der Tag- und Token-Kette anhand von bestimmten Satzzeichen, die bei der Wort-Sequenzierung immer als eigenes Token eingefügt werden, macht robuster gegen Verschiebungen im Ergebnis wie in Abschnitt 3.1.1 beschrieben: Wird Satz-weise verglichen und entstehen in den Sätzen Verschiebungen durch ungleiche Token-Zerlegungen, dann verschwindet dieser Fehler, sobald ein neues Segment betreten wird, da die neuen betrachteten Segmente wieder an gleicher Stelle beginnen.

Anreicherung mit Zusatzinformationen ist möglich, da im Gegensatz zu einer einfachen Textuellen Darstellung auf ein Token nicht exakt ein Tag folgen muss. Hier kann beispielsweise pro Token statt nur dem First-Best-Tag eine beliebig lange Liste der *N-Best* Tags stehen.

Ein solches alternatives Format ist jedoch von Operatoren, die nicht speziell dafür vorbereitet sind (wie der Evaluationsoperator), nicht lesbar. Darum müssen Ergebnisse auch noch in einem Text-Format ausgegeben werden.

3.3 Parsing

Muss bei der Evaluation ein Text-Format angenommen werden, beispielsweise das Ergebnis in einem Goldstandard, muss entweder das angenommene Format exakt bekannt und definiert sein, oder das Dokument muss auf POS-Tags durchsucht werden. Letzteres bedeutet, dass dem Parser das Tagset des angenommenen Dokuments bekannt sein muss. Hierzu muss also jedes möglicherweise auftretende Tagset definiert werden.

3.4 Evaluation

Um die Performance eines Taggers zu bewerten, müssen Metriken eingeführt werden. Diese entstehen aus Vergleichen zwischen annotierten Korpora und den Tagging-Ergebnissen. Im Rahmen dieses Projekts wird nur der Vergleich von Ergebnissen mit identischen Tagsets betrachtet. Abbildungsfunktionen zwischen Tagsets oder zu einem übergeordneten, generalisierten Tagset sind prinzipiell (mit einem gewissen Informationsverlust) möglich, werden aber hier außen vor gelassen. Betrachtet werden die folgenden Metriken zur Evaluation von Tagging-Ergebnissen:

3.4.1 Per-Tag-Accuracy

Die Per-Tag-Accuracy (ab hier nur *Accuracy*) ist die wichtigste und einfachste Form der Bewertung eines Taggers. Sie ist definiert als [C R18] :

$$\frac{\#\ korrekter\ Tags}{\#\ aller\ Token}$$

3.4.2 Per-Sentence-Accuracy

Analog zur Accuracy pro Tag kann auch Accuracy pro Satz definiert werden:

$$\frac{\#\ vollständig\ korrekter\ Sätze}{\#\ aller\ Sätze}$$

Hierzu müssen jedoch über Wort-Sequenzen hinaus auch Satz-Sequenzen definiert werden. In der Evaluation in Abschnitt 2.1 wurde dies ebenfalls berücksichtigt. :TODO

3.4.3 Confusion Matrix und daraus resultierende Metriken

Konzentrieren wir uns auf bestimmte Tags (Labels), können wir genauere Aussagen Treffen (Für allgemeine Definitionen siehe [C R18], für mehrklassige Klassifizierungsprobleme siehe [Gan14]). Zuerst bilden wir die *Confusion Matrix*. In Abbildung 3.2 befindet sich ein Beispiel dafür. In dieser Matrix wird gegenübergesetzt, wie viele Labels aus dem Goldstandard als welche Labels vom Tagger vorhergesagt wurden.

True Positives für Label X (kurz TP(X)) sind Vorhersagen, die dem Goldstandard entsprechen (in der Abbildung Gelb markiert).

False Positives (FP(X)) sind alle Vorhersagen X, die dem Goldstandard widersprechen.

True Negatives (TN(X)) sind alle Vorhersagen, die nicht X sind, und auch im Goldstandard nicht X lauten.

False Negatives (FN(X)) sind alle Vorhersagen, die nicht X sind, im Goldstandard aber X.

	GoldLabel_A	GoldLabel_B	GoldLabel_C	
Predicted_A	**30**	20	10	TotalPredicted_A=60
Predicted_B	50	**60**	10	TotalPredicted_B=120
Predicted_C	20	20	**80**	TotalPredicted_C=120
	TotalGoldLabel_A=100	TotalGoldLabel_B=100	TotalGoldLabel_C=100	

Abb. 3.2: Beispiel für eine Confusion Matrix für Labels A, B, C. Entnommen von [Gan14]

Daraus lassen sich Precision(Wie viele der Vorhersagen X waren korrekt?), Recall(Wie viele X im Goldstandard wurden korrekt erkannt?) und der F1-Score(auch F-Score, harmonisches Mittel aus Precision und Recall) berechnen:

$$Precision(X) = \frac{TP(X)}{TP(X) + FP(X)} = \frac{Korrekte\ X}{Vorhergesagte\ X}$$

$$Recall(X) = \frac{TP(X)}{TP(X) + FN(X)} = \frac{Korrekte\ X}{X\ im\ Goldstandard}$$

$$F1\text{-}Score = 2 * \frac{Precision(X) * Recall(X)}{Precision(X) + Recall(X)}$$

Für alle drei dieser Metriken steht der Wert 1 für vollständig richtig und 0 für vollständig falsch.

4 Implementierung

Dieses Kapitel behandelt die Implementierung der RapidMiner-Erweiterung *postagger*. Zuerst wird grob auf den technischen Rahmen seitens RapidMiner eingegangen, dann werden im Abschnitt 4.2 Ziele der Implementierung angesprochen.
Nach einer Übersicht in Abschnitt 4.3 über die Komponenten, die Funktionen aus Kap. 3 implementieren, werden die Komponenten einzeln im Detail vorgestellt.

4.1 RapidMiner

:TODO(Aufbau auf Text Processing!) :TODO(Large Lizenz)

Die Datenverarbeitungs-Plattform RapidMiner (kurz *RM*) ist in Java geschrieben, und Erweiterungen dafür ebenfalls. RapidMiners Code ist Open Source und Erweiterungen basierend auf dort definierten Strukturen sind erwünscht. RapidMiners zentrale Funktionalität ist es, verschiedene Funktionen (*Operatoren*) in einem Modell hintereinanderzuschalten und mit ihnen Daten zu extrahieren und mit verschiedensten Mitteln zu transformieren oder zu untersuchen. Ein Beispiel-Prozess von RapidMiner findet sich in Abbildung :TODO(Bild:bsp-Prozess).

Operatoren funktionieren entfernter betrachtet wie Funktionen in herkömmlichem Code: Sie nehmen beliebig viele Eingaben mittels *InputPorts* und *Parametern* entgegen, führen eine Aufgabe aus und haben mindestens eine Ausgabe in Form von *OutputPorts*. Alle Operatoren erben von Superklasse *Operator*. Eine Erweiterung fügt in der Regel solche Operatoren hinzu.

IO-Objekte bzw. IOObjects sind die Datenformate im RM-Modell. Sie werden von der Prozesswurzel sowie von Operatoren ausgegeben und können von Operatoren sowie dem Prozess-Ende entgegengenommen werden. Sie erben von der Klasse *IOObject* und können ebenfalls von Erweiterungen hinzugefügt werden. Operatoren können für ihre Inputs definieren, welche Typ von IOObject sie verlangen und das Modell ist nicht lauffähig, wenn diese Spezifikation nicht eingehalten wird.

Für eine ausführliche Dokumentation von RM siehe :NC(Doku), für eine Anleitung zum erstellen von Erweiterungen :NC(your own Extension)

4.2 Ziele

Diese Erweiterung soll u.A. vom auftraggebenden Lehrstuhl weiterverwendet und -entwickelt werden können. Das heißt, eine gute Dokumentation und Strukturierung der Komponenten steht im Vordergrund. Da die Erweiterung nicht unbedingt von Programmierern benutzt werden soll, muss außerdem auf eine niedrige Fehleranfälligkeit der Implementierung geachtet werden.

4.2.1 Erweiterbarkeit

4.2.2 Robustheit

4.3 Übersicht

:TODO neues Diagramm!

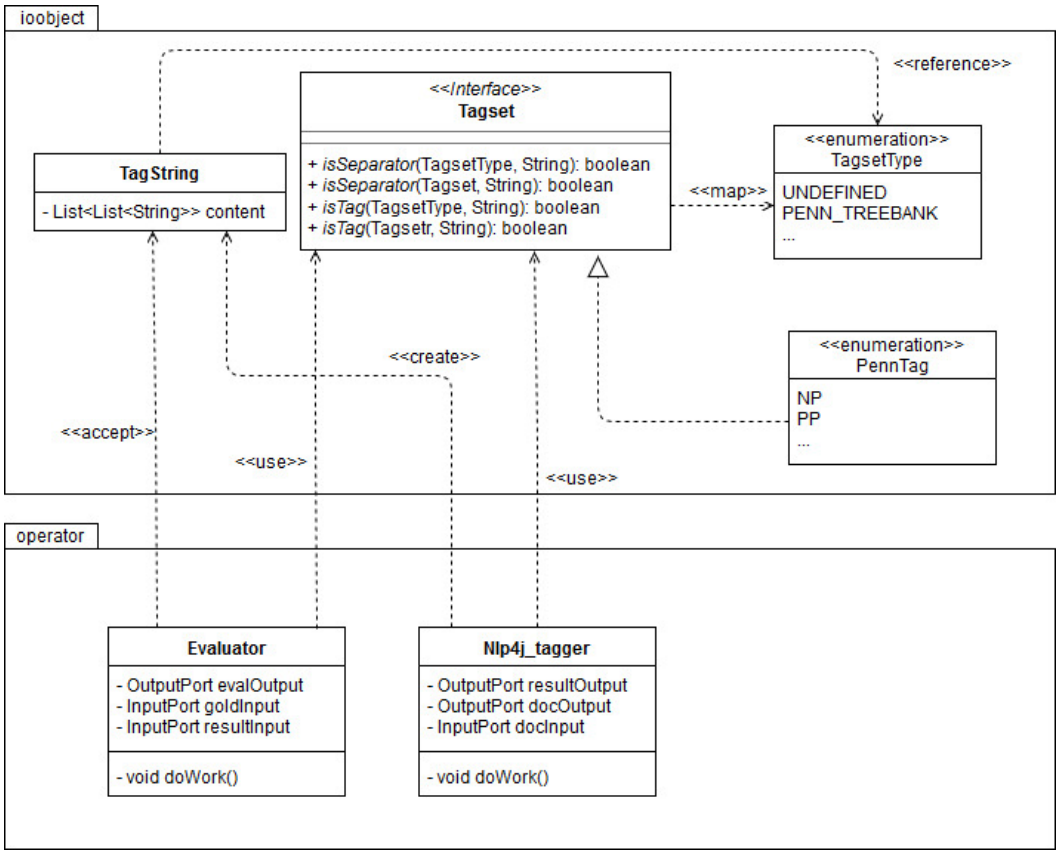

Abb. 4.1: gekürztes Klassendiagramm

Abbildung 4.1 zeigt einen Überblick über die implementierten Klassen. Um das Diagramm überschaubar zu halten, wurden nur die Klassen eingezeichnet, die zusätzlich zum Erweiterungs-Template von RapidMiner :NC implementiert wurden. Außerdem wurden nur die notwendigen Enumerationen und Klassen für den NLP4J-Tagger aufgenommen, da andere Tagger analog strukturiert sind. Es folgt eine kurze Erklärung der verschiedenen Komponenten:

Operatoren: *Evaluator* und *Nlp4j_ tagger* sowie alle anderen Klassen, die von der RapidMiner-Klasse *Operator* erben, sind auch in RapidMiner als Operatoren vertreten. Alle Operatoren außer Evaluator beinhalten einen POS-Tagger und überbrücken sowohl Ein- als auch Ausgangsformate zwischen dem externen RapidMiner-Modell (*Document* oder *TagString*) und dem Tagging-Algorithmus selbst. Der Evaluator berechnet die Unterschiedlichkeit zwischen zwei Ergebnissen, wobei eines der beiden bestenfalls vollständig korrekt ist. :TODO

Tagsets: Das Interface Tagset muss von jeder Tag-Enumeration, wie zum Beispiel dem der Penn-Treebank implementiert werden. Zusätzlich bietet es statische Methoden, die die Werte der Enumeration TagsetType auf die korrespondierenden Enumerationen abbilden, sowie Abfragen über deren Werte anbieten (hierzu wird der Typ sowie ein Tag verlangt). Die implementierenden Tagsets führen alle Part-of-Speech-Tags auf und liefern zusätzlich die Information, ob das Tag in der Datenstruktur *TagString* (s.u.) Zeilen abbricht. Auf diese Weise ist es leicht, das Projekt um ein neues Set zu erweitern.

TagString: Diese Klasse bietet, wie in Abschnitt **??** angesprochen, ein einheitliches Format für POS-Tags. Sie beschreibt den Tagset-Typen sowie die Zahl an N-Besten Tags pro Token (1 für First-Best) und strukturiert die Tags in Zeilen, die immer dann enden, wenn das letzte Tag ein *Separator* ist. :TODO

4.4 Tagsets

4.5 Eingabe und Tokenization

4.6 Ergebnisformat

4.7 Implementierte Tagger

4.8 Evaluations-Operator

Um Ergebnisse hinsichtlich ihrer Qualität zu prüfen, muss mit Goldstandards wie z.B. dem WSJ-Corpus :NC verglichen werden. Der Evaluationsoperator muss die Formate dieser Standards annehmen und interpretieren können. Neben dem eigentlichen Evaluieren ist also auch das *Parsen* eingehender Texte eine wichtige Aufgabe des Operators.

4.8.1 Parsing

Verschiedene Formate kodieren POS-Tags unterschiedlich. Eine einfache Notation ist die bereits bekannte Variante, bei der nach jedem Token ein „\“ und dann das Tag folgt. Andere Notationen kodieren zusätzlich Satzstrukturinformationen, sind also mittels Klammern geschachtelt, um z.B. Teilsätze zu markieren. Hier gibt es kein Symbol, das eindeutig ein POS-Tag ankündigt. Um die Parser-Methode des Evaluationsoperators präziser arbeiten können zu lassen, wurde ein Parameter hinzugefügt, in dem man das Format wählen kann. Es wurden zwei Modi für den Parser implementiert:

Backslash-Notation: In der oben genannten Notation, in der ausschließlich POS via „\“ markiert werden, ist Parsing einheitlich. Der Text wird an jedem Leerzeichen gespalten und jeder entstehende Substring am „\“. Das POS-Tag findet sich dann im zweiten String, der bei der zweiten Spaltung entsteht.

"None": Falls die Notation unbekannt ist, versucht der Parser, den Text an allen sinnvollen Symbolen, insbesondere dem Leerzeichen zu Trennen. Alle Substrings werden dann überprüft, ob sie ein POS-Tag sind.

Zusätzlich kann die Option *„Ignore Brackets “* gewählt werden, falls Klammern für die Formatierung des einzulesenden Textes verwendet wurden. diese werden dann ignoriert.
Als Ergebnis gibt der Parser das Format TagString aus, das dann weiter verwendet werden kann. Der Tagset-Typ des TagStrings muss via Parameter angegeben werden.

4.9 Conclusion

5 Evaluation

Die folgenden POS-Tagger wurden implementiert:

Tagger (Training Korpus)	Performance (Korpus)
NLP4J	:TODO
LingPipe	:TODO
FastTag	-

Tab. 5.1: Liste der Implementierten Tagger. Performance laut eigenen Angaben.
* : Performance aus :NC

Die Performance in Tab. 5.1 bezieht sich auf eigene Angaben der Entwickler. Mit dem implementierten Evaluationsoperator sind wir allerdings in der Lage, die Tagger an einem eigenen annotierten Goldstandard-Korpus zu testen. Die folgenden Abschnitte diskutieren die Auswahl eines passenden Korpus und die Performance der Tagger auf diesem.

5.1 Goldstandard-Korpus

Ein annotierter Korpus muss dem Tagset der einzelnen Tagger entsprechen. Dies erweist sich als schwierig, da jeder Tagger u.U. Zeichen in unterschiedliche Token-Ketten spaltet und anders Kodiert. Es macht also Sinn, den Korpus und seine Tags auf den Tagger anzupassen, sofern das möglich ist, ohne den Informationsgehalt des Korpus zu ändern.

5.2 Concepts Section 2

5.3 Concepts Section 3

5.4 Conclusion

6 Conclusion

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Das hier ist der zweite Absatz. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

6.1 System Section 1

Und nun folgt – ob man es glaubt oder nicht – der dritte Absatz. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie

„Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Nach diesem vierten Absatz beginnen wir eine neue Zählung. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

6.2 System Section 2

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander

stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Das hier ist der zweite Absatz. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Und nun folgt – ob man es glaubt oder nicht – der dritte Absatz. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst

viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

6.3 Future Work

Nach diesem vierten Absatz beginnen wir eine neue Zählung. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“?

Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Literatur

[C R18] Venkat N. Gudivada C. R. Rao. *Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications*. Elsevier Science, 2018 (zitiert auf den Seiten 1, 10).

[MP 93] B. Santorini M.P. Marcus M. A. Marcinkiewicz. „Building a large annotated corpus of English: the penn treebank“. In: *Computational Linguistics - Special issue on using large corpora: II* Volume 19 Issue 2, June 1993 (1993), S. 313–330 (zitiert auf Seite 3).

[Par07] P. Paroubek. „Evaluating Part-of-Speech-Tagging and Parsing“. In: *Evaluation of Text and Speech Systems. Speech and Language Technology, vol. 37*. Springer, Dordrecht, 2007. Kap. 4 (zitiert auf Seite 5).

[Smi11] Noah Smith. *Linguistic Structure Prediction*. Morgan und Claypool Publishers, 2011 (zitiert auf den Seiten 1, 2, 7).

Websites

[Gan14] Kavita Ganesan. *Computing Precision and Recall for Multi-Class Classification Problems*. 2014. URL: `http://text-analytics101.rxnlp.com/2014/10/computing-precision-and-recall-for.html` (besucht am 16. Apr. 2019) (zitiert auf den Seiten 10, 11).

[Sta03] Stanford University. *Penn Treebank P.O.S. Tags*. 2003. URL: `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html` (besucht am 4. Apr. 2019) (zitiert auf Seite 3).

Abbildungsverzeichnis

Tabellenverzeichnis

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig verfasst und dabei keine anderen als die angegebenen Hilfsmittel benutzt habe. Sämtliche Stellen der Arbeit, die im Wortlaut oder dem Sinn nach Publikationen oder Vorträgen anderer Autoren entnommen sind, habe ich als solche kenntlich gemacht. Die Arbeit wurde bisher weder gesamt noch in Teilen einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Bayreuth, 23.April 2019

Philipp Scholz