

Fabian Boehlke

Die Auswertung von Radiotranskripten des Norddeutschen Rundfunks (NDR) mithilfe der Programmiersprache Python

Ein kurzer Projektbericht aus dem Sommersemester 2022

TYP DES DOKUMENTS | TYPE OF THE DOCUMENT

Zeitschriftenartikel / Journal Article

Nachnutzung | Reuse

Diese Publikation steht unter der Creative-Commons-Lizenz Namensnennung 4.0 International (CC BY 4.0 International). Sofern die Namen der Autor*innen/ Rechteinhaber*innen genannt werden, kann der Inhalt vervielfältigt, verbreitet, öffentlich aufgeführt und kommerziell genutzt werden. Außerdem dürfen Bearbeitungen angefertigt und verbreitet werden. Weitere Informationen und die vollständigen Bedingungen der Lizenz finden Sie hier: <https://creativecommons.org/licenses/by/4.0/deed.de>.

Zeitschriftenartikel

Begutachtet

Begutachtet:Dr. Steffen Rudolph HAW Hamburg
Deutschland**Erhalten:** 13. Dezember 2022**Akzeptiert:** 18. Januar 2023**Publiziert:** 31. Januar 2023**Copyright:**

© Fabian Boehlke.

*Dieses Werk steht unter der Lizenz
Creative Commons Namens-
nennung 4.0 International (CC BY 4.0).***Empfohlene Zitierung:**

BOEHLKE, Fabian, 2023: Die Auswertung von Radiotranskripten des Norddeutschen Rundfunks (NDR) mithilfe der Programmiersprache Python. Ein kurzer Projektbericht aus dem Sommersemester 2022. In: *API Magazin* 4(1) [Online] Verfügbar unter: [DOI 10.15460/apimagazin.2023.4.1.137](https://doi.org/10.15460/apimagazin.2023.4.1.137)

Die Auswertung von Radiotranskripten des Norddeutschen Rundfunks (NDR) mithilfe der Programmiersprache Python

Ein kurzer Projektbericht aus dem Sommersemester 2022

Fabian Boehlke^{1*} ¹ Hochschule für Angewandte Wissenschaften Hamburg, Deutschland

Student im 5. Semester des Studiengangs Bibliotheks- und Informationsmanagement

* Korrespondenz: redaktion-api@haw-hamburg.de

Zusammenfassung

Die Programmiersprache Python ist multifunktional. Unter anderem lassen sich mit ihr große Textmengen sprachlich und inhaltlich auswerten. Im Rahmen des Seminars Wissensorganisation 2 an der Hochschule für Angewandte Wissenschaften Hamburg im Sommersemester 2022 sollten wir auf Grundlage von Python einen selbst zusammengestellten Textkorpus in zwei Teilprojekten inhaltlich auswerten und klassifizieren. Im Folgenden werden beide Teilprojekte kurz vorgestellt.

Schlagwörter: Python, Radiobeitrag, Transkript, Textkorpus, Inhaltsanalyse, Klassifikation

The evaluation of radio transcripts of the Norddeutscher Rundfunk (NDR) by using the Python programming language

A short project report from the summer term 2022

Abstract

The Python programming language is multifunctional. Among other things, it can be used to evaluate the language and content of large amounts of text. In the context of the seminar Knowledge Organization 2 at Hamburg University of Applied Sciences in the summer term 2022, we were to evaluate and classify the content of a self-compiled text corpus in two subprojects on the basis of Python. In the following, both subprojects are briefly presented.

Keywords: Python, Radio Report, Transcript, Text Corpus, Content Analysis, Classification

1 Einführung

Entstanden ist das vorliegende Projekt im Rahmen des Seminars Wissensorganisation 2 an der Hochschule für Angewandte Wissenschaften Hamburg im Sommersemesters 2022.¹ Nach den Einführungen in die Grundlagen von Python bestand unsere Prüfungsleistung darin, einen selbst erstellten Textkorpus mit Hilfe der Programmiersprache Python inhaltlich und strukturell auszuwerten. Da ich als Werkstudent im Schallarchiv des Norddeutschen Rundfunks (NDR) tätig bin, fiel meine Wahl recht schnell auf die Auswertung von Radiotranskripten aus der Hörfunkdatenbank des NDR, zu der ich freien Zugang habe.

1.1 Textkorpus

Für die Erstellung meines individuellen Textkorpus' recherchierte ich über die Suchfunktion der Hörfunkdatenbank nach Beiträgen (siehe Abb. 1). Hierfür legte ich insgesamt sechs Suchwörter fest: „Literatur“, „Ausstellung“, „Konzert“, „Biografie“, „Tourismus“ und „Sport“. Mit diesen Suchwörtern habe ich über das Eingabefenster „Überall suchen“ in der Hörfunkdatenbank recherchiert. Aus den Suchergebnissen wählte ich jeweils zwölf Texte zu jedem Suchbegriff aus, so dass mein Korpus am Ende 72 Texte umfasst hat. Die Auswahl erfolgte dabei rein subjektiv nach meinem eigenen Empfinden, welche Radiobeiträge ich für geeignet hielt. Dies muss bei der späteren Analyse berücksichtigt werden.

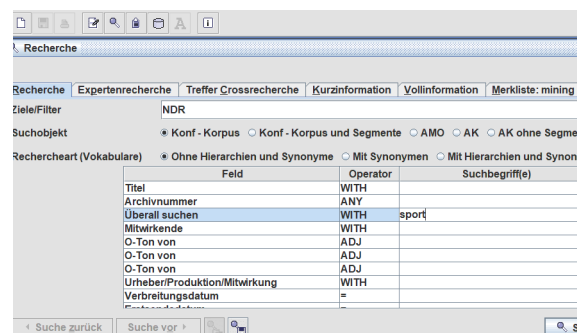
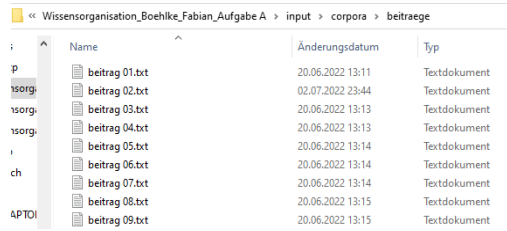


Abb. 1: Suchfunktion in der Hörfunkdatenbank des NDR (Eigener Screenshot)

Die 72 Texte wurden von mir als Textdateien im txt-Dateiformat abgespeichert – dies ist Voraussetzung für die Auswertung mit Python (siehe Abb. 2). Dabei ergänzte ich der Vollständigkeit halber bei jedem Beitrag den Namen des Autors bzw. der Autorin, das Sendedatum sowie den jeweiligen Radiosender. Zu den Radiosendern gehörten u.a. NDR 90,3, NDR Kultur, NDR Info sowie weitere Lokalsender der norddeutschen Bundesländer. Die Texte wurden einzeln benannt – hier beitrag 01, beitrag 02 etc. – und im Input-Ordner der Projektdatei abgespeichert. In den Input-Ordner kamen alle Daten sowie weitere Dateien, auf welche im Rahmen der Arbeit mit Python zugegriffen werden musste.

1 Eine Veröffentlichung der originalen Projektdokumentation ist auf dem studentischen Publikationsportal SA/IL der HAW Hamburg geplant, erreichbar unter: <https://reposit.haw-hamburg.de/sail> [Online, Zugriff am 13.12.2022].



Name	Änderungsdatum	Typ
beitrag 01.txt	20.06.2022 13:11	Textdokument
beitrag 02.txt	02.07.2022 23:44	Textdokument
beitrag 03.txt	20.06.2022 13:13	Textdokument
beitrag 04.txt	20.06.2022 13:13	Textdokument
beitrag 05.txt	20.06.2022 13:14	Textdokument
beitrag 06.txt	20.06.2022 13:14	Textdokument
beitrag 07.txt	20.06.2022 13:14	Textdokument
beitrag 08.txt	20.06.2022 13:15	Textdokument
beitrag 09.txt	20.06.2022 13:15	Textdokument

Abb. 2: Beiträge als Textdateien (Eigener Screenshot).

Zur näheren inhaltlichen Auseinandersetzung war es Aufgabe, bereits vor der automatischen Analyse mindestens zehn Texte manuell auszuwerten und mit jeweils drei Schlagwörtern zu versehen. In meinem Fall waren es zwölf Texte, jeweils zwei für jeden Suchbegriff.

1.2 Technische Umsetzung

Die Anwendung von Python wurde mit Hilfe der Arbeitsumgebung Jupyter Notebook2 realisiert, welche in der Distribution Anaconda enthalten und als Grundvariante kostenlos verfügbar ist. Anaconda ist speziell für die wissenschaftliche Anwendung von Python gedacht und gut zur Verarbeitung großer Datenmengen und zur Visualisierung geeignet. Allerdings sind die Anwendungen von Anaconda nicht immer auf die aktuellste Version von Python eingebunden (Kofler 2022, S. 384 f.). Das Jupyter Notebook besteht aus einzelnen Zellen, welche beliebig lang gefüllt werden können (siehe Abb. 3). Der Code kann so pro einzelner Zelle ausgeführt werden, was für den Anfang von Vorteil ist. Es muss darauf geachtet werden, dass die Befehle in unteren Zellen von weiter oben stehendem Code abhängig sind, welcher zuvor ausgeführt sein muss. Sofern der Befehl eine Aktion beinhaltet, findet sich das Ergebnis direkt unter der Zelle. Es ist sinnvoll, zu Beginn des Skripts die Bibliotheken einzuziehen. Eine Python-Bibliothek versammelt Funktionen und Anwendungen, welche ansonsten erst umfangreich im Skript erstellt werden müssten (Dürkop 2020). Daten, die ausgewertet oder eingebunden werden sollen, lassen sich über den Input-Ordner importieren. Dies betrifft die genannten Textdateien ebenso wie weitere Werkzeuge, die etwa mit Excel erstellt wurden. Die Skripte können zur weiteren Bearbeitung als IPYNB-Dateien abgespeichert werden.



```
In [53]: print(pd.__version__)
1.3.4

In [54]: #Zusammenfassung der einzelnen Schritte der Sortierung der Tokens nach Häufigkeit im Korpus
def tokensNachHäufigkeitSortiert(tokens, columnName):
    frequent = FreqDist(tokens)
    anzahlTokens2 = len(tokens)
    most_frequent = frequent.most_common(anzahlTokens2)
    index = [tup[0] for tup in most_frequent]
    tokens_sortiertNachHäufigkeiten = pandas.DataFrame([tup[1] for tup in most_frequent], columns=[columnName], index = index)
    return(tokens_sortiertNachHäufigkeiten)

#Zusammenfassung der einzelnen Schritte der Sortierung der Tokens nach Häufigkeit in einzelnen Dokumenten
def tokensNachHäufigkeitSortiert_dokument(tokens, columnName):
    frequent = FreqDist(tokens_dokument)
    anzahlTokens2 = len(tokens_dokument)
    most_frequent = frequent.most_common(anzahlTokens2)
    index = [tup[0] for tup in most_frequent]
    tokens_sortiertNachHäufigkeiten_dokument = pandas.DataFrame([tup[1] for tup in most_frequent], columns=[columnName], index = index)
    return(tokens_sortiertNachHäufigkeiten_dokument)
```

Abb. 3: Ausschnitt aus einem Jupyter-Notebook (Eigener Screenshot)

2 <https://jupyter.org/> [Online, Zugriff am 13.12.2022].

Die Prüfungsleistung des Kurses bestand aus zwei Teilprojekten, einem Indexierungs- und einem Klassifikationsprojekt, auf die im Folgenden näher eingegangen wird.

2 Teilprojekt 1: Indexierung

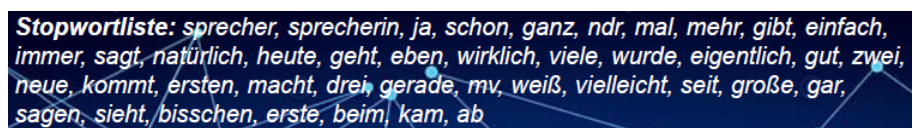
2.1 Fragestellung

Wie oben beschrieben, erfolgte die Zusammenstellung meines Textkorpus subjektiv. Im Rahmen meines Indexierungsprojektes wollte ich nun genau diese Zusammenstellung überprüfen und meinen Rechercheweg kritisch hinterfragen. Daher lautete die Fragestellung, welche Relevanz meine ausgewählten Suchwörter tatsächlich für die Texte in dem Korpus haben. Daneben gab es weitere kleinere Aufgaben, etwa sollten die zwölf Texte, die zuvor individuell ausgewertet wurden auffindbar gemacht werden. Gleiches galt für die fünf häufigsten Wörter des Textkorpus.

2.2 Stoppwortliste

Um die inhaltliche Analyse des Textkorpus zu verbessern, ist es von Vorteil, als vorbereitende Maßnahme eine Liste mit Stoppwörtern (stopwords) anzulegen (siehe Abb. 4). Stoppwörter sind Wörter, die bei der anschließenden Textanalyse nicht berücksichtigt werden, da sie keine inhaltliche Relevanz besitzen. So lässt sich vermeiden, dass einem bei der Textanalyse häufig vorkommende, aber bedeutungslose Substantive, Adverbien, Pronomen, Artikel oder Ähnliches angezeigt werden.

Für die Erstellung der Stoppwortliste ließ ich mir mit Hilfe von Python die häufigsten Wörter in meinem Textkorpus anzeigen und traf daraufhin eine Auswahl. Die Wörter listete ich in Excel auf, diese Excel-Liste wurde anschließend im Input-Ordner abgespeichert.



Stoppwortliste: *sprecher, sprecherin, ja, schon, ganz, ndr, mal, mehr, gibt, einfach, immer, sagt, natürlich, heute, geht, eben, wirklich, viele, wurde, eigentlich, gut, zwei, neue, kommt, ersten, macht, drei, gerade, mv, weiß, vielleicht, seit, große, gar, sagen, sieht, bisschen, erste, beim, kam, ab*

Abb. 4: Liste meiner Stoppwörter (Eigene Darstellung)

2.3 Thesaurus

Eine weitere vorbereitende Maßnahme war die Erstellung eines Thesaurus. Hintergrund ist, dass die Wörter im Korpus einfacher auffindbar gemacht werden sollen. Durch die unterschiedlichen Formen eines Wortes, die in Texten verwendet werden – bspw. streik, streiken, streikende –, wäre eine automatische Suche sehr eingeschränkt. Ein Weg, dieses Problem zu lösen, ist das „Stemming“, also die Reduktion des Wortes auf seinen Wortstamm.

Im Falle meines Projektes besteht der Thesaurus aus einer Tabelle mit fünf Spalten. Die erste Spalte (`deskriptor_stemming`) beinhaltet den Stamm des Wortes, in der zweiten Spalte (`nicht_deskriptor`) findet sich der Token, also die Zeichenkette in Wortform, welche auf den Wortstamm zurückgeführt werden soll. Die dritte Spalte (`deskriptor_kontrolliert`) enthält ein intellektuell vergebenes Schlagwort, welches in den meisten Fällen aber dem Begriff entspricht, so wie er im Text vorkommt. In der vierten Spalte (`oberbegriff`) findet sich ein Oberbegriff, welchen ich zumeist auf Grundlage der ARD-Sachklassifikation vergeben habe. In der fünften und letzten Spalte (`begrueudung`) findet sich entweder der Quellennachweis oder eine kurze Erläuterung zur Auswahl der Begriffe (siehe Abb. 5).

	deskriptor_stemming	nicht_deskriptor	deskriptor_kontrolliert	oberbegriff	begrueudung
1	barschel	barschel	Barschel, Uwe	ehem. Ministerpräsident von Schleswig-Holstein	
2	krimis	krimis	Kriminalroman	Literarisches Werk	ARD-Sachklassifikation
3	ministerpräsident	ministerpräsident	Ministerpräsident	Regierungschef	Synonym von Regierungschef/ARD-Sachklassifikation
4	ministerpräsident	ministerpräsidenten	Ministerpräsident	Regierungschef	Mehrzahl von Ministerpräsident/Synonym/ARD-Sachklassifikation
5	ministerpräsidentin	ministerpräsidentin	Ministerpräsidentin	Regierungschef	Weibliche Form von Ministerpräsident/ARD-Sachklassifikation
6	heimatroman	heimatromane	Heimatroman	Literarisches Werk	ARD-Sachklassifikation
7	nationalsozialismus	nationalsozialismus	Nationalsozialismus	Geschichte des nationalsozialis	ARD-Sachklassifikation
8	nationalsozialist	nationalsozialist	Nationalsozialist	Geschichte des nationalsozialis	ARD-Sachklassifikation
9	elbmarsch	elbmarsch	Elbmarschen	Marschland	Marschland an der Unterelbe
10	wanderausstell	wanderausstellung	Wanderausstellung	Ausstellung mit wechselndem Ort	
11	ausstell	ausstellung	Ausstellung	Museumswesen	ARD-Sachklassifikation
12	ausstell	aussteller	Ausstellung	Museumswesen	ARD-Sachklassifikation
13	ausstell	ausstellungen	Ausstellung	Museumswesen	Plural von Ausstellung/ARD-Sachklassifikation
14	herd	herd	Herd	Haushaltsgerät	ARD-Sachklassifikation
15	herd	herde	Herd	Haushaltsgerät	Plural von Herd/ARD-Sachklassifikation

Abb. 5: Ausschnitt meines Thesaurus in Excel (Eigener Screenshot)

Für den Thesaurus berücksichtigte ich einerseits die Schlagwörter, welche ich für die zwölf erwähnten, manuell ausgewerteten Texte vergeben habe. Im Thesaurus lassen sich ebenfalls die sechs Suchbegriffe finden, auf deren Grundlage der Korpus erstellt wurde, sowie die fünf häufigsten Begriffe im Korpus. Gleiches gilt für Wortvarianten, die im Korpus vorkommen und auf demselben Wortstamm basieren, wie andere Begriffe im Thesaurus.

2.4 Umsetzung

Bei der Umsetzung in Python wird – nach Einzug der notwendigen Bibliotheken – zunächst der Korpus selbst geladen, indem der Dateipfad zum Input-Ordner angegeben wird. Anschließend werden die Stoppwörter entfernt, die eigene Stoppwortliste wird hierzu ebenfalls aus dem Input-Ordner geladen. Auch der Thesaurus wird aus dem Input-Ordner geladen, damit Suchabfragen auf Grundlage dieses Vokabulars gestartet werden können. Für die Jupyter Notebooks konnten wir zumeist auf Vorlagen zurückgreifen, so dass wir bei der Umsetzung der Projekte diese nur anpassen mussten. Die Suchfunktion auf Grundlage des Thesaurus machte ich mir dann für die Klärung meiner Fragestellung zu Nutze.

Ich wendete die Suchfunktionen auf all meine sechs Suchbegriffe an und konnte so feststellen, wie oft jedes der Suchwörter in den zwölf Texten, auf deren Grundlage sie recherchiert worden sind, jeweils vorkommt. Die Suchfunktion, die auf Grundlage des Thesaurus basiert, macht es beispielsweise auch möglich, sich den Kontext des jeweiligen Suchbegriffes anzeigen zu lassen. Voraussetzung ist natürlich, dass der Suchbegriff im Thesaurus vermerkt ist. Entsprechend der Spalteneinteilung ist es auch möglich, nach Wortstämmen oder Oberbegriffen zu suchen. Im zweiten Schritt

meiner Umsetzung berechnete ich dann für die Suchbegriffe sowohl die Termfrequenz (tf) sowie die Termfrequenz*Inverse Document Frequency (tf*idf).

Bei der Termfrequenz handelt es sich um die relative Häufigkeit eines Wortes in einem Text. Berechnet wird sie, indem man die Häufigkeit des Wortes im Text durch die Gesamtzahl der Wörter im Text teilt. Die Inverse Document Frequency (idf) gibt die Bedeutung eines Wortes für den gesamten Textkorpus an. Ihre Berechnung erfolgt, indem die Gesamtzahl der Dokumente im Korpus durch die Zahl der Texte geteilt wird, in welchen der untersuchte Begriff vorkommt. Tf*idf bildet die Kombination aus beidem. Die Gewichtung eines Begriffes ist umso höher, je größer seine relative Häufigkeit bezogen auf ein Dokument ist und/oder umso seltener der Begriff im Korpus auftaucht. Jeder Begriff hat pro Text einen individuellen tf*idf-Wert ([vgl. Rajaraman/Ulman 2011](#)).

Bei der Berechnung der Termfrequenz war für mich der Wert des jeweiligen Suchbegriffes relevant. Gleiches galt für die Berechnung von tf*idf, wobei ich die Berechnung hier zweimal vorgenommen habe. Einmal habe ich mir anzeigen lassen, wie relevant die einzelnen Suchbegriffe für ihre jeweiligen zwölf Texte sind. Außerdem habe ich den tf*idf-Wert jedes einzelnen Suchbegriffes bezogen auf den gesamten Korpus berechnet.

Auch hier ist der Korpus nach Themen bzw. Suchwörtern sortiert, so dass das entsprechende Thema durch Entfernung der # ausgewählt werden kann.

```
In [18]: # Die Dokumente der Sammlung zusammenfügen
#Literatur
#txtDocs = [txt1, txt2, txt3, txt4, txt5, txt6, txt7, txt8, txt9, txt10, txt11, txt12]

#Ausstellung
#txtDocs = [txt13, txt14, txt15, txt16, txt17, txt18, txt19, txt20, txt21, txt22, txt23, txt24]

#Konzert
#txtDocs = [txt25, txt26, txt27, txt28, txt29, txt30, txt31, txt32, txt33, txt34, txt35, txt36]

#Biografie
#txtDocs = [txt37, txt38, txt39, txt40, txt41, txt42, txt43, txt44, txt45, txt46, txt47, txt48]

#Tourismus
#txtDocs = [txt49, txt50, txt51, txt52, txt53, txt54, txt55, txt56, txt57, txt58, txt59, txt60]

#Sport
txtDocs = [txt61, txt62, txt63, txt64, txt65, txt66, txt67, txt68, txt69, txt70, txt71, txt72]

docs = [list(nltk.tokenize.word_tokenize(txtDoc)) for txtDoc in txtDocs]

# Mit einer For-Schleife durch jedes Dokument iterieren
for i, doc in enumerate(docs):
    print("Tf im Dokument {}: ".format(i + 1))
```

Abb. 6: Berechnung der Termfrequenz (Eigener Screenshot)

Für die Berechnung von tf sowie tf*idf müssen die jeweiligen Beiträge, auf welche sich die Kalkulation bezieht, zu Sammlungen zusammengefügt werden (siehe Abb. 6). Dies war verhältnismäßig aufwendig, da ich im Notebook jeden Beitrag einzeln auflisten musste. Die Auflistung musste einmal für tf sowie jeweils einmal für tf*idf bezogen auf die einzelnen Suchbegriffe und auf den gesamten Korpus durchgeführt werden.

2.5 Ergebnisse

Mit Hilfe der Suchfunktion wollte ich überprüfen, wie oft jeder der einzelnen Suchbegriffe in den jeweils zwölf dafür ausgewählten Texten vorkommt. Hier hat die Untersuchung ergeben, dass sowohl „Ausstellung“, „Konzert“, „Tourismus“ und

„Sport“ in all ihren Texten vorkommen. Der Suchbegriff „Biographie“ taucht nur in sieben von zwölf der für diesen Suchbegriff ausgewählten Texte auf, „Literatur“ sogar nur in einem der zwölf Texte (siehe Abb. 7).

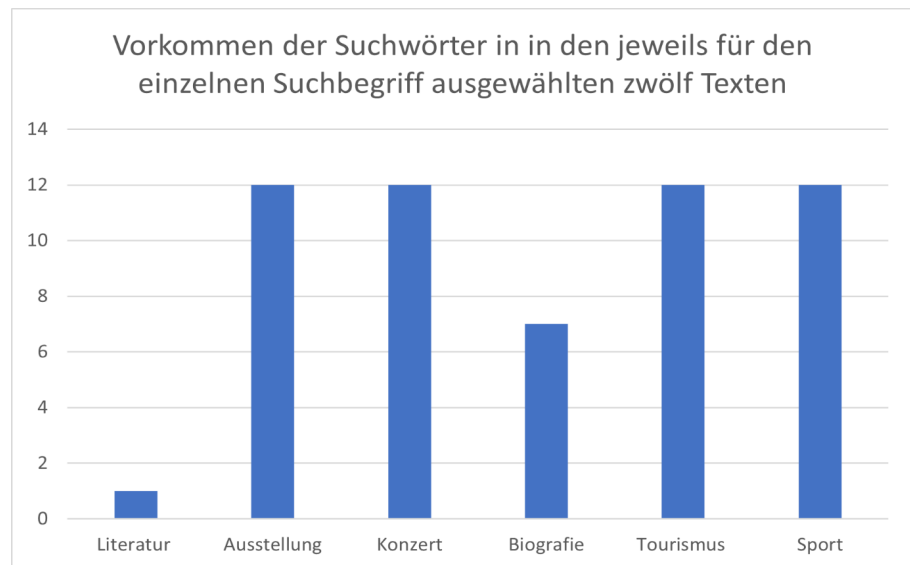


Abb. 7: Vorkommen der jeweiligen Suchwörter in den jeweils für den einzelnen Suchbegriff ausgewählten zwölf Texten (Eigene Darstellung)

Die Auswertung der Berechnung der tf- sowie tf*idf-Werte war kleinteilig und aufwendig. Da es für jeden der 72 Beiträge eigene Werte gibt, jeweils einen für die Termfrequenz und zwei für tf*idf, da die Berechnung hier doppelt erfolgen musste, waren konkrete Rückschlüsse schwierig. Muster waren insoweit erkennbar, dass der tf*idf-Wert bei den Begriffen „Ausstellung“, „Konzert“, „Tourismus“ und Sport“ bezogen auf die eigenen zwölf Texte jeweils bei 0,0 lag. Hintergrund ist die schon geteilte Erkenntnis, dass die Begriffe jeweils in all ihren Texten vorkommen und genau deshalb im Verhältnis zur Zahl ihrer Texte keine große Relevanz haben. Im Falle der tf*idf-Werte lässt sich sagen, dass je größer der Wert ist, desto höher ist auch die Relevanz des jeweiligen Suchwortes. Auf den ersten Blick lässt sich dies schnell anhand der Nullen hinter dem Komma feststellen. Bei nur einer Null hat der Suchbegriff eine hohe Relevanz, bei mehr als einer Null ist die Relevanz geringer (siehe Tabelle 1).

Als Fazit aus dem ersten Teilprojekt lässt sich festhalten, dass insbesondere die Auswertung der Häufigkeit des Vorkommens der Suchbegriffe in ihren jeweiligen Texten weiter geführt hat. Die Übersicht der Häufigkeiten hat einen klaren Überblick über die Bedeutung der einzelnen Suchbegriffe gegeben. Die Berechnung der tf sowie tf*idf-Werte hat diese Bedeutung zwar bestätigt, mir aber ansonsten keine neuen Erkenntnisse geliefert, da die systematische Auswertung zu kleinteilig wäre.

Tabelle 1: Beispiel für die tf- und tf*idf-Werte für die Suchbegriffe in unterschiedlichen Beiträgen

Suchwort/ Beitrag	tf	tf*idf (Themenkomplex)	tf*idf (Gesamter Korpus)
„Literatur“			
Beitrag 01	0.0031847133757961785	0.007913715445184715	0.010121190542509382
„Ausstellung“			
Beitrag 13	0.011673151750972763	0.0	0.019116055907402296
Beitrag 20	0.00423728813559322	0.0	0.0069390202940711725
„Konzert“			
Beitrag 25	0.008097165991902834	0.0	0.012178764346366593
Beitrag 36	0.01639344262295082	0.0	0.024657006504529087
„Biografie“			
Beitrag 41	0.00425531914893617	0.002293602130777391	0.009349891818451998
Beitrag 46	0.010169491525423728	0.005481320346434104	0.022344656718673417
„Tourismus“			
Beitrag 49	0.0036101083032490976	0.0	0.006179482893698623
Beitrag 60	0.010238907849829351	0.0	0.017526110186565035
„Sport“			
Beitrag 61	0.009900990099009901	0.0	0.015530850672414309
Beitrag 72	0.008130081300813009	0.0	0.012752974942388987

Die Suchbegriffe „Ausstellung“, „Konzert“, „Tourismus“ und „Sport“ haben sich als gut geeignet herausgestellt. Sie kommen in all ihren Texten vor, die Texte sind gleichzeitig thematisch sehr eng eingegrenzt. Anders ist es bei „Literatur“ und „Biografie“. Beide Begriffe kommen in ihren Texten kaum bzw. nur teilweise vor. Der

Grund dafür dürfte sein, dass beide Begriffe nicht so spezifisch wie die anderen Suchwörter sind. Ein Radiobeitrag zum Thema Literatur kann ein oder mehrere Bücher behandeln, eine Literaturkritik beinhalten oder ein Genre oder einzelnen Autoren oder Autorinnen zum Thema haben. Ähnlich ist es mit „Biografie“. Der Suchbegriff kann sich im Radiokontext auf Porträts oder Nachrufe beziehen, auf die Besprechung literarischer Biografien oder auf einen historischen Beitrag. Ihr Auffinden im Zuge der Recherche dürfte somit eher mit der Verschlagwortung in der Hörfunkdatenbank zu tun haben als mit der tatsächlichen Relevanz des Begriffes im Radiobeitrag selbst. Hier wäre es im Wiederholungsfall sinnvoll, auf andere Begriffe zurückzugreifen, welche die Thematik besser eingrenzen, beispielsweise Rezension, Literaturkritik oder Porträt.

3 Teilprojekt 2: Klassifizierung

3.1 Kontext und Fragestellung

Im Seminar wurden zwei unterschiedliche Möglichkeiten der Klassifikation vorgestellt, die statische Klassifikation und die lernenden Systeme. Da ich meinen Textkorpus mit Hilfe der sechs Suchbegriffe zusammengestellt hatte, bot es sich an zu überprüfen, ob es möglich ist, die Texte dem richtigen Suchbegriff zuzuordnen. Einerseits war diese Fragestellung eine gute Fortführung des ersten Teilprojektes, da es auch hier um die Qualität der Suchbegriffe ging. Andererseits ließen sich so beide genannten Klassifikationssysteme miteinander vergleichen.

3.2 Statische Klassifikation

Das Verfahren der statischen Klassifikation ist relativ begrenzt und einfach aufgebaut. Grundsätzlich besteht es daraus, dass dem System eine Reihe an Begriffen gegeben werden, die jeweils zu einer der Kategorien bzw. Suchbegriffen gehören (siehe Abb. 8). Anschließend berechnet das System, wie viele Übereinstimmung es gibt. Mit Hilfe der if/else-Klausel in Python erfolgt eine Zuordnung durch das System (siehe Abb. 9). Die Zuordnung kann eindeutig sein, das System kann aber auch sagen, dass eine eindeutige Zuordnung nicht möglich ist.

2) Listen für die Zuordnung zu den Kategorien Literatur und Biografie

```
In [775]: # Liste Wörter Literatur. In die Liste wurden nur Wörter aufgenommen,
# die in den Beitragstexten zum Thema Literatur mindestens 7 Mal vorkommen, aber keine Eigennamen darstellen.
literatur_words=['roman', 'buch', 'ddr', 'vater', 'autor', 'geschichte', 'dorf', 'spielen', 'sohn', 'fast', 'erzählt', 'liebe', 'nacht']
print(literatur_words)

['roman', 'buch', 'ddr', 'vater', 'autor', 'geschichte', 'dorf', 'spielen', 'sohn', 'fast', 'erzählt', 'liebe', 'nacht']

In [776]: # Liste Wörter Biografie. In die Liste wurden nur Wörter aufgenommen,
# die in den Beitragstexten zum Thema Biografie mindestens 7 Mal vorkommen und keine Eigennamen darstellen.
biografie_words=['biografie', 'später', 'deutschland', 'frau', 'niemand', 'lassen', 'familie', 'ab']
print(biografie_words)

['biografie', 'später', 'deutschland', 'frau', 'niemand', 'lassen', 'familie', 'ab']
```

Abb. 8: Begriffe für die Zuordnung im Rahmen der statischen Klassifikation mit Python (Eigener Screenshot)

Da die statische Klassifikation sehr eingeschränkt ist, habe ich in diesem Fall nur die beiden Suchbegriffe „Literatur“ und „Biografie“ miteinander verglichen. Die beiden Begriffe habe ich deshalb gewählt, weil es auch durchaus Überschneidungen geben kann, etwa in Form einer literarischen Biografie, was die Zuordnung interessanter macht, als wenn man beispielsweise zwei völlig unterschiedliche Bereiche wie Literatur und Sport ausgewählt hätte.

Zuordnung des Beitrages

```
Mit Hilfe einer if/else-Klausel werden die berechneten Zahlen zugeordnet.

In [787]: if literatur > biografie:
print("Der Beitrag ist vermutlich aus der Kategorie Literatur.")
elif biografie > literatur:
print("Der Beitrag ist vermutlich aus der Kategorie Biografie.")
else:
print("Eindeutig zuordnen kann man den Beitrag nicht.")

Der Beitrag ist vermutlich aus der Kategorie Biografie.

In [ ]:
```

Abb. 9: Zuordnung der Beiträge mit Hilfe der if/else-Klausel (Eigener Screenshot)

Um die Zahl der Vergleichstexte zu erhöhen, habe ich für die statische Klassifikation noch zwei weitere kleine Korpora mit jeweils zehn Texten erstellt. Hierfür recherchierte ich – wie auch für den großen Korpus – in der NDR-Hörfunkdatenbank mit den Suchbegriffen „Literatur“ und „Biografie“. Um passende Begriffe für die Zuordnung zu finden, habe ich mir jeweils die häufigsten Wörter aus den Textsammlungen „Literatur“ und „Biografie“ anzeigen lassen und in beiden Fällen rund zehn Begriffe ausgewählt. Eigennamen habe ich dabei nicht berücksichtigt.

3.3 Lernende Systeme

Das lernende bzw. dynamische Verfahren habe ich mit allen sechs Suchbegriffen durchgeführt. Beim lernenden Verfahren wird dem System ein Trainingsbestand gegeben. Für diesen Trainingsbestand habe ich zwölf der 72 Texte verwendet, jeweils zwei jedes Suchbegriffes. Aus jedem Text wird eine Zahl an relevanten Begriffen aufgelistet, getrennt durch ein Komma folgt die entsprechende Kategorie. Dies kann entweder innerhalb des Python-Codes im Jupyter Notebook erfolgen (siehe Abb. 10). Alternativ kann die Liste auch mit Excel erstellt werden.

```
In [115]:
#Den Trainingsbestand habe ich aus den 12 Texten erstellt, die ich auch genauer analysiert
#Allerdings sind die Worte des Trainingsbestandes verändert, da ich diese auf die Thematik

train = [
    ('roman krimi fiktiv', 'literatur'),
    ('heimatroman roman gegenwartsliteratur', 'literatur'),
    ('ausstellung bestaunen besucher wanderausstellung', 'ausstellung'),
    ('ausstellung museum besuch fotos', 'ausstellung'),
    ('symphoniker tournee orchester musiker', 'konzert'),
    ('philharmoniker musikfestival orchester', 'konzert'),
    ('karriere schauspieler frau ausnahmefigur', 'biografie'),
    ('porträt biografie moderatorin', 'biografie'),
    ('urlauber urlaub tourismus', 'tourismus'),
    ('jugendherberge klassenfahrt reise tourismus', 'tourismus'),
    ('liga football sport', 'sport'),
    ('fußball liga fußballverband sport', 'sport')
]
```

Abb. 10: Trainingsbestand für das lernende System im Jupyter Notebook (Eigener Screenshot)

Für die Zuordnung im lernenden Verfahren wird der jeweilige Beitrag einzeln eingelesen. Anschließend gibt das System die Zuordnung aus (siehe Abb. 11 u. 12).

```
In [131]:
# Anwendung des Classifiers auf ein neues Dokument
classifyDokument = dokuEinlesen('literatur 01.txt')
Antwort = c12.classify(textBereinigen(classifyDokument, tabelle_eigeneStopwords))
print("-----")
print("\nDas Genre des Artikels ist: " + (Antwort)) # Einordnugn in die Klasse

Datei: literatur 01.txt
-----
Literarische Vielfalt aus dem Gastland Kanada
Sprecher 1 Zum Beispiel der Journalist Michel Jean der in seinem Roman Kuckum die Geschichte seiner Urgroßmutter
```

Abb. 11: Einzug des Dokuments im lernenden System in Python (Eigener Screenshot)

```
Sprecher 9
Peter Mücke 19.10.2021 NDR Info

-----

Das Genre des Artikels ist: literatur
```

Abb. 12: Output im Jupyter Notebook: Zuordnung des Beitrags im lernenden System (Eigener Screenshot)

3.4 Ergebnis

Bei der statischen Klassifikation konnte das System 15 von 24 Beiträgen richtig zuordnen. Bei vier Beiträgen war eine eindeutige Zuordnung nicht möglich – drei aus dem Bereich „Biografie“ und einer aus dem Bereich „Literatur“. Fünf Beiträge wurden falsch zugeordnet, alle aus dem Bereich „Biografie“. Aus den beiden neuen Korpora wurden 14 Texte richtig zugeordnet, bei zweien war eine eindeutige Zuordnung nicht möglich. Vier Texte wurden falsch zugeordnet, auch hier alle aus dem Bereich „Biografie“. Die Texte aus diesem Bereich sind also am wenigsten eindeutig zu klassifizieren.

Beim dynamischen Verfahren wurden 51 von 60 überprüften Texten aus allen Themenbereichen richtig zugeordnet. Falsch zugeordnet wurden neun Texte aus den Bereichen „Literatur“ und „Konzert“. Fast alle falsch zugeordneten Texte hat das System dem Bereich „Biografie“ zugeordnet. Die beiden Zusatz-Korpora habe ich im Anschluss ebenfalls mit dem dynamischen Verfahren überprüft. Hier wurden vom System 13 der 20 Texte richtig zugeordnet, sieben hingegen falsch.

Beim Vergleich der beiden Klassifikationssysteme (siehe Abb. 13) hat ganz klar das

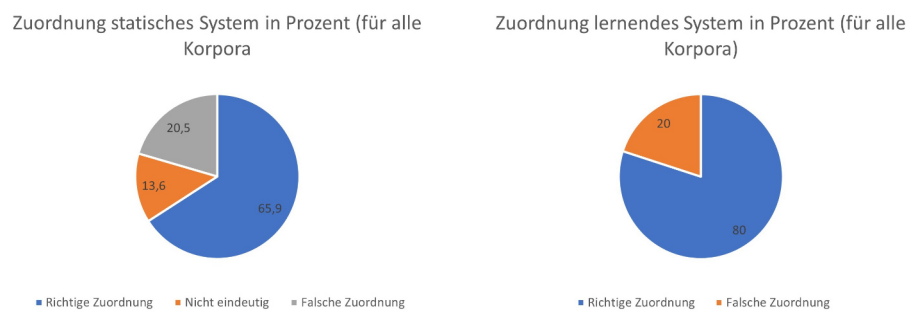


Abb. 13: Vergleich der Zuordnung statisches und lernendes System (Eigene Darstellung)

dynamische Verfahren überwogen. Bei diesem Verfahren hat das System 80 Prozent der Texte richtig zugeordnet, bei der statischen Klassifikation waren es hingegen nur knappe 66 Prozent der Texte. Einschränkend muss natürlich gesagt werden, dass die Verfahren unterschiedlich angewendet wurden und auch die Wahl des jeweiligen Klassifikationsvokabulars subjektiv gefällt wurde.

Insgesamt hat die Klassifikation aber die Ergebnisse aus dem ersten Teilprojekt untermauert. Insbesondere die Suchbegriffe „Biografie“ und „Literatur“ haben sich als sehr uneindeutig und schwer fassbar gezeigt. Besonders Texte aus diesen beiden Bereichen waren schwer zu klassifizieren. Dem Themenbereich „Biografie“ wurden zahlreiche Texte fälschlich zugeordnet.

4 Gesamtfazit

Bei Python handelt es sich um eine vielseitig einsetzbare Programmiersprache, diese Erkenntnis konnte ich aus dem Seminar mitnehmen. Durch den Input der

Lernveranstaltung sowie die praktische Anwendung in beiden Teilprojekten haben sich mir viele neue Bereiche erschlossen.

Mit der Wahl meiner Fragestellung bin ich auch im Nachgang sehr zufrieden. Zwar hat – wie beschrieben – nicht jede der Umsetzungen zu einem nutzbaren Ergebnis geführt, aber auch das ist eine Erkenntnis eines solchen Projekts. Die technische Umsetzung hat gut funktioniert, obgleich ich mir vieles erst neu erschließen musste. Hilfreich war hierbei, dass wir auf vorhandene Vorlagen zurückgreifen konnten. An einigen Punkten gestaltete sich die Umsetzung aufwendig (bspw. tf*idf) oder funktionierte nicht so, wie vorgesehen (bspw. lernende Systeme). Kleine Probleme konnte ich auch immer wieder mit Hilfe eigener Internetrecherche lösen.

Insgesamt war die Arbeit mit Python für mich ein großes Lernprojekt, welches mir auch künftig in unterschiedlichen Kontexten dienlich sein wird.

Literatur

DÜRKOP, Axel 2020. Bibliotheken verwenden. Hamburg: Technische Universität Hamburg Harburg.,12.11.2020 [Zugriff am 30.11.2022]. Verfügbar unter: <https://www3.tuhh.de/itbh/informatik-202021/veranstaltungsskript/python/processingpy/programmiergrundlagen/bibliotheken/>

KOFLER, Michael 2022. *Python: Der Grundkurs*. 2. aktualisierte Auflage. Bonn: Rheinwerk Verlag. ISBN 978-3-8362-8513-1

RAJARAMAN, Anand und ULLMAN, Jeffrey David 2011. *Mining of Massive Datasets*. Cambridge: Cambridge University Press. [Zugriff am 30.11.2022]. PDF E-Book. ISBN 978-1-1390-5845-2. Verfügbar unter: DOI: [10.1017/CBO9781139058452](https://doi.org/10.1017/CBO9781139058452)