Die Verwendung von männlichen, weiblichen und neutralen Substantiven in Bundestagsreden

Hausarbeit

in der Veranstaltung Programmieren I
im Studiengang Digital Humanities, M.Sc.

Universität Trier
FB II - Sprach-, Literatur- und Medienwissenschaften

Betreuer: Prof. Dr. Christof Schöch

Vorgelegt im September 2021 von:

Luisa Schmidt
Karl-Grün-Straße 8
54292 Trier
E-Mail: s2lascmi@uni-trier.de
Matr.-Nr. 1375568

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Lehrer und Lehrerin, LehrerIn, Lehrkraft, Lehrende: geschlechtergerechte Sprache kann auf die vielfältigste Art und Weise formuliert werden. Spielten diese Begriffe vor wenigen Jahrzehnten noch kaum eine Rolle in der deutschen Sprache und Gesellschaft, so findet man sie heute in vielen Texten, Reden und Fernseh- oder Radiobeiträgen. Dies verdeutlicht, wie präsent das Thema der geschlechtersensiblen Sprache in den letzten Jahren geworden ist.
Dabei ist geschlechtergerechte Sprache keineswegs ein neues Phänomen. Bereits 1980 legten Sprachwissenschaftlerinnen die „Richtlinien zur Vermeidung sexistischen Sprachgebrauchs“ vor und gründeten damit eine Bewegung, die heute viel Zuspruch bekommt, aber auch auf Widerstand trifft. Die Einen möchten eine Sprachkultur herstellen, in der sich Personen jedes Geschlechts gleichermaßen repräsentiert fühlen. Die Anderen, so auch Teile des Vereins Deutsche Sprache (VDS), halten geschlechtersensible Sprache für einen „zerstörerischen Eingriff in die deutsche Sprache“ (Maron et al., 2019).
Auch im Deutschen Bundestag, der gesetzgebenden Gewalt und eines der wichtigsten Diskursforen in Deutschland, ist die Verwendung geschlechterinklusiver oder -neutraler Sprache, wie die geschlechtergerechte Sprache auch genannt wird, angekommen. Ziel dieser Ausarbeitung ist es, die Verwendung geschlechtergerechter Sprache in Bundestagsreden in ausgewählten Wahlperioden automatisiert zu untersuchen. Dabei soll analysiert werden, inwiefern sich die Verwendung männlicher, weiblicher und neutraler Begriffsformen in ihrer Häufigkeit im Laufe der Wahlperioden verändert hat. Außerdem wird betrachtet, ob ein Zusammenhang zwischen der Nutzung der Begriffsformen und Geschlecht der Redenden oder deren Parteizugehörigkeit besteht. Insgesamt knüpft die vorliegende Ausarbeitung somit an die Erkenntnisse der Studie von Stecker et al. (2021) an und möchte diese bestätigen oder widerlegen.
In den folgenden Kapiteln werden zunächst die theoretischen Grundlagen erläutert, auf denen die Hausarbeit basiert. Danach werden die Forschungsfragen eingeführt. In Kapitel 4 werden die Anforderungen an den Algorithmus, vorbereitende Schritte und schließlich der entwickelte Algorithmus vorgestellt, welcher der Analyse zugrunde liegt. Dann werden die Ergebnisse der Analyse präsentiert, bevor zuletzt das eigene Vorgehen im Kapitel 6 kritisch reflektiert und bewertet wird.

2. Theoretische Grundlagen

In diesem Kapitel werden zunächst die theoretischen Grundlagen der geschlechtergerechten Sprache und die menschliche Wahrnehmung in Bezug auf die verwendete Sprache eingeleitet. Weitergehend wird ein kurzer Blick auf die Rolle von Frauen in der deutschen Politik geworfen, als auch auf ihre Präsenz in politischer Sprache.

2.1 Geschlechtergerechte Sprache und Wahrnehmung

2018 waren rund 51% aller Deutschen Frauen und stellten damit die Mehrheit der Bevölkerung dar (Bundeszentrale für politische Bildung, 2020). Dennoch sind Frauen in der deutschen Sprache deutlich seltener repräsentiert als Männer, da häufig das generische Maskulinum verwendet wird (Bundesministerium für Bildung, Wissenschaft und Forschung Österreich, 2018, S. 2). Dieses findet im Deutschen Verwendung, wenn Männer und Frauen in gleichen Teilen angesprochen werden sollen oder wenn das Geschlecht der handelnden Person nicht bekannt ist (Knoke, 2017).
Diese Unterrepräsentation von Frauen ist nicht nur fest in der Sprache verankert, sie hat auch Auswirkungen auf die gesellschaftliche Wahrnemung von Frauen. Bezugnehmend darauf legen Stecker et al. dar: „The dominance of male generics (...) contributes to an androcentric world-view, marginalizes women and proliferates gender stereotypes“ (2021, S. 2). So hat Sprache einen großen Effekt auf die Wahrnehmung von Männern und Frauen, hat aber auch gleichzeitig die Macht, diese Wahrnehmung zu verändern (ebd.). Kunze sieht „Sprache als Schlüsselwerkzeug für die Veränderung bestehender Geschlechternormen“ und bewertet daher die Verwendung „geschlechtergerechte[r] Formulierungen (...) [als] unumgänglich“ (2015, S. 3).
Der Unterrepräsentation von Frauen in Sprache soll die *geschlechtergerechte Sprache* entgegenwirken. Die bereits 1980 veröffentlichten „Richtlinien zur Vermeidung sexistischen Sprachgebrauchs“ prangerten den asymmetrischen Gebrauch von männlichen und weiblichen Substantiven in der deutschen Sprache an („Geschlechtergerechte Sprache“, Wikipedia, 2021). Die Richtlinien werden häufig als Grundstein der geschlechterneutralen Sprache in Deutschland gesehen, welche „einen Sprachgebrauch [bezeichnet], der (...) die Gleichbehandlung von Frauen und Männern (...) zum Ziel hat“ (ebd.). Die *gendergerechte Sprache* kann als Erweiterung geschlechtergerechter Sprache gesehen werden. Hier werden auch nicht-binäre Personen über die Verwendung eines Gendersterns * oder des Doppelpunktes : in die Sprache mit einbezogen. Sie fungieren als symbolischer Platzhalter für nicht-binäre Personen.
Doch die Verwendung von geschlechtergerechter Sprache ist nicht unumstritten. Payr argumentiert, dass Sprache zum Großteil auf eigenen Interpretationen basiert und

oft auf persönlichen Zuschreibungen gegründet ist (2021, S. 63). So sei es die Interpretation bestimmter Formulierungen letztendlich allen selbst überlassen (ebd.). In anderen Sprachkulturen hat sich zudem eine entgegengesetzte Bewegung durchgesetzt: Die Briten, beispielsweise, versuchen, „das Anzeigen von Geschlechtlichkeit so weit wie möglich zu vermeiden“ (Pollatschek, 2020) und verwenden das generische Maskulinum. Auch in Frankreich und Italien bestehen ähnliche Ansichten. Darüber hinaus wirft Payr die Frage auf, ob „der Rückgriff auf die gleiche Form, die auch Männer benutzen, nicht sogar den Gedanken der Gleichberechtigung besser zum Ausdruck [bringt] als eine spezielle weibliche Form“ (2021, S. 61).

2.2 Frauen in der deutschen Politik

Die Politik war für lange Zeit männerdominiert, Frauen nahmen kaum am politischen Leben teil. Im ersten Deutschen Bundestag lag der Frauenanteil zu Beginn der Wahlperiode bei lediglich 6,8%, nur 28 der insgesamt 410 Mitglieder waren Frauen (Bundeszentrale für politische Bildung, 2017; „Frauenanteil im Deutschen Bundestag seit 1949“, Wikipedia, 2021). In der Gesamtbevölkerung hingegen stellten 1950 die Frauen in der BRD die Mehrheit, mit 25,3 Millionen im Gegensatz zu 22,4 Millionen Männern (Rahlf, 2015, S. 32).
Um die Teilhabe von Frauen in der Politik zu stärken, haben z.B. Die Grünen eine innerparteiliche Frauenquoten-Regelung eingeführt. So wird Frauen bei der Besetzung höherer Ämter der Vortritt gegeben. Außerdem vergeben Die Grünen ungerade Listenplätze nur an Frauen, während die geraden Plätze an beide Geschlechter gehen dürfen (Grüne Köln, k.D.). Auch die SPD und DIE LINKE schreiben sich einen Frauenanteil von 40% bzw. 50% für politische Ämter und Mandate vor, während CDU/CSU, FDP und AfD keine Quoten verfolgen (Höhne, 2020).
Trotz dieser Regelungen sind Frauen im Bundestag gegenüber ihrem Anteil an der Gesamtbevölkerung immernoch unterrepräsentiert. Zu Beginn des aktuellen 19. Deutschen Bundestags waren 30,9% der Abgeordneten Frauen. Den höchsten Frauenanteil erreichte der 18. Bundestag zu Ende der Wahlperiode mit 36,3% (Bundeszentrale für politische Bildung, 2017).

2.3 Frauen in politischer Sprache

Das Parlament reflektiert den Sprachgebrauch eines Landes, kann ihn aber auch bedeutend mitgestalten (Stecker et al., 2021, S. 2). So können auch im Diskurs des Deutschen Bundestags sprachliche Entwicklungen erkannt werden. Aus historischer Perspektive gesehen hatten Die Grünen einen großen Einfluss auf die Sprache im Bundestag: Nach ihrer erstmaligen Wahl ins Parlament erhöhte sich nicht nur der

Anteil weiblicher Abgeordneten im Parlament deutlich aufgrund der innerparteilichen Quotenregelung, Die Grünen revolutionierten zudem, wie Frauen in die politische Sprache aufgenommen und in ihr repräsentiert wurden (ebd., S. 3). Insgesamt haben die Parteien des linken Spektrums sehr früh damit begonnen, geschlechtergerechte Sprache in der politischen Arena zu etablieren (ebd.).
Dabei spielen in der Politik laut Stecker et al. besonders die Strategien der Neutralisierung und Feminisierung eine herausragende Rolle, um geschlechtergerechter zu formulieren (2021, S. 3). Bei der Neutralisierung werden Substantive verwendet, die kein Geschlecht haben oder anzeigen. So soll durch „die Beseitigung diskriminierender Ausdrücke“ (Wesian, 2007, S. 20) eine weniger geschlechter-spezifische Sprache etabliert werden. Als Beispiele können hier die neutrale Formulierung *Arbeitskraft* oder die Substantivierung *Arbeitende* genannt werden. Die Feminisierung hingegen zielt auf die „sprachliche Sichbarmachung (...) von Frauen mittels Geschlechterspezifikation“ (ebd., S. 17) ab. Es werden weibliche Substantive verwendet oder zu den männlichen zu ergänzt (ebd.). Unter Feminisierung fallen damit also Formulierungen wie *Arbeiterinnen und Arbeiter*.

2.4 Geschlechtergerechte Sprache im Deutschen Bundestag

Stecker, Müller, Blätte und Leonhardt analysieren in der quantitativen Studie „The evolution of gender-inclusive language. Evidence from the German Bundestag, 1949-2021“ (2021) die Nutzung geschlechtergerechter Sprache im Bundestag. Der verwendete Korpus enthält alle Reden, die seit der Eröffnung des Bundestags 1949 gehalten wurden (ebd., S. 4). Automatisiert untersucht wird die Verwendung männlicher und weiblicher Substantive anhand von symmetrischen Wortpaaren in Singular und Plural, wie z.B. Politiker/Politikerin (ebd.).
Die Ergebnisse der Studie zeigen, dass die Verwendung weiblicher Begriffsformen über die Perioden hinweg zugenommen hat (Stecker et al., 2021, S. 1). Zudem stellen die Forscher einen Zusammenhang zwischen Geschlecht sowie Partei der Redenden und der Verwendung weiblicher Substantive fest: Weibliche Abgeordnete tendieren häufiger dazu, weibliche Begriffsformen in Debatten zu nutzen als ihre männlichen Kollegen (ebd., S. 5). Außerdem werden weibliche Substantive häufiger von Abgeordneten des linken Spektrums verwendet als von konservativen Mitgliedern des Bundestags (ebd., S. 7). Geschlechterneutrale Begriffe wie beispielsweise *Arbeitskraft* oder *Lehrende* wurden in der Studie nicht berücksichtigt, da sie in der ersten Sichtung des Korpus nicht häufig vorkamen.

3. Forschungsfragen und Begriffsdefinition

Um die Entwicklung des Algorithmus und die Analyse der Daten sinnvoll zu strukturieren, werden drei Forschungsfragen formuliert, die dann mithilfe des Algorithmus beantwortet werden sollen. Da die vorliegende Ausarbeitung weitestgehend die Ergebnisse der Studie von Stecker et al. (2021) nachvollziehen möchte, ergeben sich die folgenden drei Forschungsfragen (FF):

FF 1: Wie hat sich die Verwendung von männlichen, weiblichen und neutralen Begriffsformen im Laufe der analysierten Wahlperioden verändert?

FF 2: Welche Unterschiede gibt es in der Verwendung von männlichen, weiblichen und neutralen Begriffsformen bezüglich des Geschlechts des Redners oder der Rednerin?

FF 3: Welche Unterschiede gibt es in der Verwendung von männlichen, weiblichen und neutralen Begriffsformen bezüglich der Fraktionszugehörigkeit des Redners oder der Rednerin?

Eine zentrale Rolle in dieser Ausarbeitung spielen die Begrifflichkeiten der *männlichen, weiblichen* und *neutralen Begriffsformen*, weshalb eine knappe Definition der Begriffe erfolgt.
Als *männliche Begriffsformen* werden Substantive gewertet, die dem männlichen Geschlecht zugeordnet werden können, also die Referenz zu einem Mann bilden. Zu diesen werden beispielsweise die Wörter Soldat, Minister oder Patienten. Im Gegensatz dazu werden Substantive, die Frauen referenzieren, als *weibliche Begriffe* gewertet. Hierunter fallen dann die Wörter Soldatinnen, Ministerin oder Patientinnen. Zuletzt sind *neutrale Begriffsformen* in dieser Ausarbeitung Begriffe, die geschlechterneutral formuliert sind und weder Männern noch Frauen allein zuzuordnen sind. Diese neutralen Begriffe sind beispielsweise Substantivierungen wie Arbeitende oder Lehrende, aber auch neutrale Formulierungen wie Ansprechperson oder Teilzeitkräfte.

4. Implementierung des Algorithmus

4.1 Anforderungen an den Algorithmus

Das Ziel der Ausarbeitung ist es, eine *automatisierte Analyse* der Verwendung von männlichen, weiblichen und neutralen Begriffsformen in verschiedenen Wahlperioden durchzuführen. Dabei soll besonders auf Parteizugehörigkeit und Geschlecht der Redenden geschaut und die Ergebnisse in übersichtlichen *Visualisierungen* dargestellt werden. Für die *Analyse* muss das Programm zunächst die Redetexte der Plenarsitzungen einlesen und auch bereinigen können, um eine Analyse zu ermöglichen. In den Schritt der Bereinigung fällt unter anderem das Entfernen einiger Satzzeichen, als auch das Entfernen von allen Textteilen, die vor dem eigentlichen Beginn und nach dem offiziellen Ende der Sitzung vermerkt sind - Vorspann und Anlagen. Die Wahlperiode, aus der die Texte stammen, sollte vermerkt werden, um Aussagen über die zeitliche Entwicklung treffen zu können. Darüber hinaus sollte das Programm die einzelnen Redetexte der Sprechenden effizient voneinander unterscheiden und extrahieren können. Hier mit inbegriffen ist das Erkennen der Fraktionszugehörigkeit sowie des Geschlechts der Abgeordneten, da diese beiden Variablen wichtig sind für die spätere Auswertung der Daten. Innerhalb der einzelnen Redetexte sollte das Programm darüber hinaus männliche, weibliche und neutrale Substantive erkennen und zählen können. Zuletzt ist es wichtig, dass die gesammelten Daten über Wahlperiode, Partei und Geschlecht der Abgeordneten sowie die Verwendung der männlichen, weiblichen und neutralen Substantive in einer Tabelle gesammelt werden.
Für die *Visualisierungen* sollte das Programm die bereitgestellten Daten einlesen und weiternutzen können. Für Letzteres sollten wichtige Daten in der Tabelle beispielsweise aufsummiert oder von absoluten in relative Häufigkeiten umgerechnet werden, um die Basis für aussagekräftige Ergebnisse zu legen. Zudem sollten die Ergebnisse der Forschungsfragen in übersichtlichen Grafiken dargestellt werden und die Beantwortung der Fragen ermöglichen.

4.2 Vorbereitende Schritte

Zunächst müssen die Plenarprotokolle heruntergeladen werden. Hierfür bietet der Deutsche Bundestag einen Service an, der alle Protokolle jeder Wahlperiode zur Verfügung stellt (siehe „Open Data“, Deutscher Bundestag). Die Protokolle sind in XML ausgezeichnet.
In der Ausarbeitung werden insgesamt vier Perioden analysiert. Bei Dokumenten zu

den Perioden 1, 7 und 14 handelt es sich meist um reine Redeprotokolle, denen einige XML-Tags hinzugefügt wurden, um ein valides XML-Dokument zu erstellen. Es sind innerhalb des Textes also kaum Auszeichnungen vorzufinden, was eine manuelle Extraktion der gesuchten Informationen bedingt. Die Protokolle der 19. Periode sind hingegen umfangreich ausgezeichnet, sodass sie mit der Bibliothek `Beautiful Soup` bearbeitet und nach den benötigten Informationen durchsucht werden können (siehe Richardson, k.D.).
Für die Zählung von männlichen, weiblichen und neutralen Wörtern in den Reden muss jeweils eine Vergleichsliste erstellt werden. Diese nutzt der Algorithmus, um Strings zu erkennen und zuzuordnen. Für die Liste männlicher und weiblicher Substantive werden symmetrische Wortpaare wie Lehrer/Lehrerin verwendet, sodass die beiden Listen analog zueinander sind. Für die neutralen Begriffe werden einige Wörter aus den anderen beiden Listen in ihrer neutralen Form genommen, z.B. Lehrkraft. Es werden aber auch weitere Begriffe hinzugefügt, die nicht in anderer Form in den Listen zu männlichen und weiblichen Begriffsformen vorkommen.
Da Geschlecht und Partei der Redenden in der Analyse eine zentrale Rolle spielen, müssen diese aus dem Redetext extrahiert werden. Deshalb wird eine Liste aller weiblichen Abgeordneten sowie zwei Listen aller Parteien erstellt, die in den relevanten Perioden im Bundestag waren. Für die Parteien werden zwei Listen erstellt, da der Algorithmus mit `Beautiful Soup` die Parteien der 19. Periode anhand des XML-Tags `<fraktion>` erkennt. Die manuelle Extraktion hingegen erkennt die Partei anhand der Klammern um den Parteinamen und den folgenden Doppelpunkt: *(SPD):*. Alle genannten Vergleichslisten sind im Anhang A hinterlegt.

4.3 Vorstellung des Algorithmus

In diesem Kapitel wird der Algorithmus präsentiert. Der vollständige Programmcode kann im Anhang B oder unter `https://github.com/s2lascmi/Hausarbeit_ProgrammierenI` eingesehen werden. Der Umfang der Erläuterungen richtet sich nach Komplexität und Neuheit der Funktion. In den ersten Zeilen des Programms finden sich die benötigten Importe, darauf folgen die geschriebenen Funktionen.

4.3.1 Extraktion der Daten

Die erste Funktion des Algorithmus `def read_list` liest eine Datei ein und bereinigt sie. Bei den eingelesenen txt-Dateien handelt es sich um die in Kapitel 4.2 angesprochenen Listen mit weiblichen Abgeordneten `[female_MPs]` und den jeweiligen Substantiven `[female_words]`, `[male_words]`, `[neutral_words]`. Außerdem werden die Listen für die Extraktion der Partei `[parties]` bzw. `[parties_for_soup]` eingelesen (Z. 542-547) und als Python-Liste zurückgegeben.

Die nächste Funktion `def read_textfile` liest die XML-Plenarprotokolle ein. Die folgende `def clean_texts` nutzt als Parameter diese eingelesenen Texte und bereinigt sie, wie bereits in Kapitel 4.1 beschrieben. Für fast jede Periode müssen diese Entfernungen einzeln definiert werden, da sich die Eröffnungs- und Schließungsphrasen der Sitzungen über die Perioden hin verändert haben und somit ein einzelnes `re.sub` nicht ausreicht.
Es folgen die Funktionen, in denen die einzelnen Redetexte, Geschlecht und Parteizugehörigkeit der Sprechenden extrahiert werden. Hier wurde jeweils mit zwei verschiedenen Funktionen gearbeitet, eine für die 19. und eine für die übrigen Perioden. Dies ist mit der unterschiedlich detaillierten XML-Auszeichnung der Daten in den Perioden zu begründen, weshalb eine Funktion alleine nicht ausreicht. Da bei den für Periode 19 angewendeten Funktionen mit der Bibliothek `Beautiful Soup` gearbeitet wurde, sind diese Funktionen mit dem Zusatz `soup` benannt.
`def split_text_new_speaker` nutzt als Parameter die bereinigten Texte der Perioden 1, 7 und 14. Zunächst wird eine leere Liste erstellt, welche dann nach folgendem Schema befüllt wird (Z. 63-123):

Abbildung 4.1: Auschnitt aus der Funktion `split_text_new_speaker`

```
64      find_all_speakers_CDU1 = re.finditer("\(CDU\):", cleansed)
65      for item in find_all_speakers_CDU1:
66          list_search_results.append(item.end())
```

Je Parteibezeichnung wird über die Methode `re.finditer` im Text ein String gesucht. In der for-Schleife wird dann für jeden gefundenen String die Position seines letzen Zeichens in die Liste hinzugefügt. In den drei letzten Zeilen der Funktion wird zur Liste die Gesamtlänge des Textes hinzugefügt, die Liste aufsteigend sortiert und im return-Statement zurückgegeben. Eine Kombination der Suchbegriffe über Boolesche Operatoren ist hier nicht zuverlässig möglich, weshalb die Strings einzeln gesucht werden.
Die folgende Funktion `def find_start_of_line` verwendet die Parameter `cleaned_text`, also den bereinigten Text, `list_search_results`, was nach Zeile 560 die Liste von Positionen ist, die in der Funktion `def split_text_new_speaker` erstellt wurde, und die Laufvariable `i`. Ziel der Funktion ist es, den Beginn des Redetextes einer einzelnen Person zu finden. Dies geschieht über die Suche nach dem ersten Zeilenumbruch vor dem Parteikürzel, dessen Position im Text in der vorherigen Funktion gefunden wurde. Das Ergebnis der Funktion ist die Extraktion der einzelnen Sprechertexte, welche ab dem gefundenen Zeilenumbruch mit dem Namen der Abgeordneten beginnen, die Partei einschließen und dann bis zum nächsten Sprechertext gehen. Zunächst wird die Variable `beginning_of_line` gleichgesetzt mit dem Wert aus der Liste mit Index `i` (Z. 132). Es folgt ein if-Statement welches testet,

ob der Wert von `beginning_of_line` der Länge des analysierten Textes entspricht, also ob gerade der letzte Wert in der Liste verwendet wird. Ist diese Bedingung erfüllt, wird die Funktion direkt verlassen, da ansonsten der Index out of range liefe. Wenn nicht, wird die Variable `begin_found = False` gesetzt (Z. 136). Nun springt die Variable `str` in der while-Schleife an die Position im zu analysierenden Text, welche über das Element in der Liste vorgegeben wird. Im if-Statement wird geprüft, ob sich an der Position im String ein Zeilenumbruch befindet. Falls ja, hat der Algorithmus den Beginn der Zeile gefunden und verlässt die Schleife. Ansonsten prüft das Programm das else-Statement, in welchem eine Position weiter vorne im analysierten String auf den Zeilenumbruch geprüft wird. Schließlich wird die Variable `name_and_party` verwendet, welche Namen und Partei der Sprechenden enthält. `whole_speaker_text` beinhaltet den Sprechertext inklusive Name und Partei bis hin zum Beginn des nächsten Sprechers. Beide Variablen werden zurückgegeben.

Abbildung 4.2: Die Funktion `find_start_of_line`

```
131 def find_start_of_line(cleaned_text, list_search_results, i):
132     beginning_of_line = list_search_results[i]

134     if len(cleaned_text) == beginning_of_line:
135         return
136     begin_found = False
137     while not begin_found:
138         str = cleaned_text[beginning_of_line]
139         if str == "\n":
140             begin_found = True
141         else:
142             beginning_of_line = beginning_of_line - 1

145     name_and_party = cleaned_text[int(beginning_of_line + 1):(int(
        list_search_results[i]))]
146     whole_speaker_text = cleaned_text[int(beginning_of_line + 1):(
        int(list_search_results[i + 1]))]
147     return [name_and_party, whole_speaker_text]
```

Die Funktion `def create_dictionary` mit den bereits erwähnten Namens-, Partei- und Substantivlisten als Paramentern folgt. Ebenso als Paramenter erhält sie die Liste `single_speakers_text`, welche Namen und Partei der Abgeordneten enthält, als auch den Sprechertext. Zuerst wird ein Dictionary initialisiert, das als Keys die Namen der Informationen verwendet, die extrahiert werden sollen (Z. 153f.). Dann wird der erste Listeneintrag aus `[single_speakers_text]` ausgewählt, welcher mit der `.split()-Methode` nochmals am Leerzeichen getrennt wird. Über `[-1]` und `[-2]` werden die letzte und vorletzte Position der Tokens ausgewählt und in if-else-Statements nach Partei und Name der Abgeordneten - über den dann das Geschlecht erfasst werden kann - durchsucht. Hier werden die Partei- und Namenslisten genutzt. Die gefundenen Personen- und Parteinamen werden als Values den

entsprechenden Keys im Dictionary zugeordnet. Zuletzt wird der Sprechertext in der Variable `word_tokens_speech` ausgewählt und nach Leerzeichen gesplitted. Neben der Gesamtwortanzahl pro Sprechertext wird auch das Vorkommen der verschiedenen Substantive gezählt und deren Anzahl in das Dictionary eingetragen. Auch hier werden die entsprechenden Listen als Abgleich genutzt, die Funktion gibt das Dictionary zurück (Z. 155-187). Einen Ausschnitt der Funktion zeigt Abbildung 4.3:

Abbildung 4.3: Auschnitt aus der Funktion `def create_dictionary`

```
169     surname = word_tokens_name_party[-2]
170     if len(word_tokens_name_party) == 1:
171         return
172     if surname in female_MPs:
173         dict_words["Gender of speaker:"] = "w"
174     else:
175         dict_words["Gender of speaker:"] = "m"
```

Nach diesem Vorgehen wird das Dictionary für die Perioden 1, 7 und 14 befüllt, dessen Index sich aus Dokumentname und einer Ziffer zusammensetzt, die für jeden Sprechertext hochgezählt wird (Z. 571). Zu Beginn jeder Periode wird zudem im Dictionary ein leerer Eintrag ergänzt, der später für die Weiterverarbeitung genutzt wird (Z. 554f.).
Die Zeilen 190-241 beinhalten Funktionen, welche mit `Beautiful Soup` die Texte aus Periode 19 bearbeiten. Zunächst werden die Texte eingelesen, bereinigt und in der Variable `soup` ein SoupObject erstellt. Ebenfalls in der Main-Funktion wird `soup` nach dem XML-Tag `<rede>` durchsucht, das den Beginn eines Sprecherbeitrags markiert. Ein Index für die einzelnen Einträge im Dictionary wird nach selbem Schema vergeben, wie oben bereits beschrieben (Z. 580-587, 599).
In einer while-Schleife, welche sich an der Laufvariable `i` und der Anzahl der `<rede>`-Tags orientiert, wird dann die eigentliche Analyse durchgeführt. Die Funktion `def extract_info_from_soup` (Z. 190-201) verwendet die gefundenen Reden als Parameter und durchsucht sie nach den Tags `<nachname>` für die Variable `speaker`, `<fraktion>` für die Variable `party` und `<p klasse=*>` für die Variable `speech`. * kann hier für die Klassen J_1, J, Z oder O stehen, welche Redetext und Zitate im Protokoll markieren (siehe Deutscher Bundestag, 2015). Die Variablen werden im return-Statement zurückgegeben. Die Kombination der Suchanfragen für `speech` ist von hoher Wichtigkeit für die Extraktion (siehe Abbildung 4.4), denn ohne die Kombination kann jeweils nur eine der Klassen extrahiert werden.

Abbildung 4.4: Auschnitt aus der Funktion `def extract_info_from_soup`

```
200     speech = single_speeches.find_all("p", {"klasse": ("J_1", "J",
        "Z", "O")})
```

`def create_dictionary_from_soup` erhält als Parameter die Partei-, Namens- und Substantivlisten und die Variable `analyse`, welche die Ergebnisse der vorherigen Funktion enthält. Die Funktion arbeitet analog zu `def create_dictionary`: Zunächst wird aus der Variable `analyse` das jeweils passende Listenelement für Name, Partei und Sprechertext extrahiert und gesplitted. Dann wird getestet, ob das vorliegende Item in der Liste zu Namen, Parteien oder Substantiven vorkommt. Das Dictionary wird befüllt und zurückgegeben (Z. 204-241).
Folgend werden die Daten in das in Zeile 539 initialisierte Dictionary eingefügt und ergänzen die Daten der anderen Perioden, bevor ein Eintrag mit dem Key „Ende“ hinzukommt (Z. 600-603). Der letzte Schritt in der Extraktion der Daten ist die Funktion `def save_dictionary_as_df`, welche das entstandene Dictionary in einen DataFrame umwandelt und als csv-Datei abspeichert. Hier wird die Bibliothek `pandas` verwendet (siehe Pandas Development Team, k.D.).

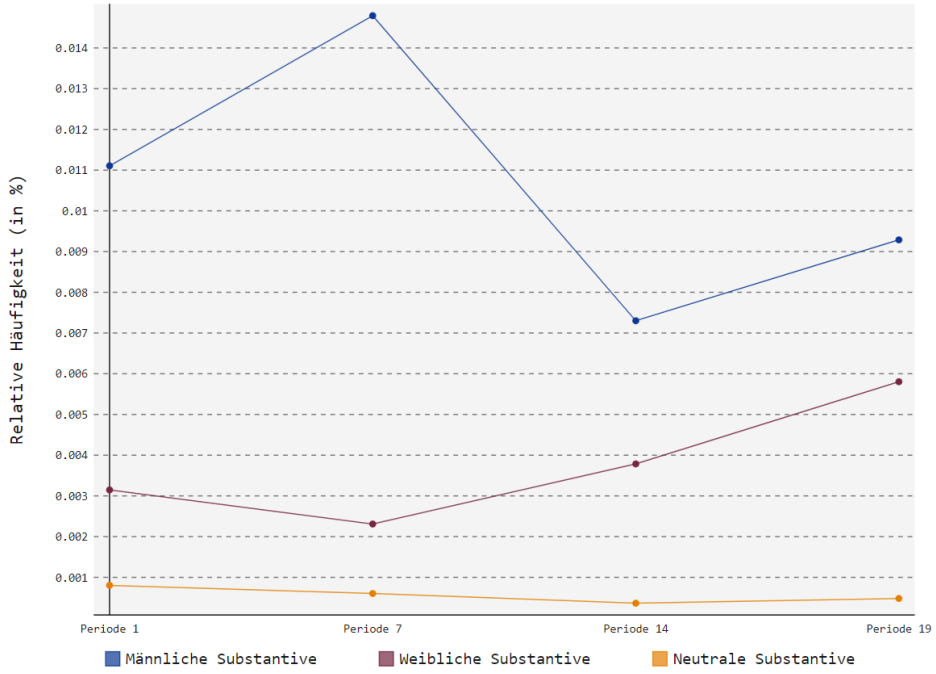
4.3.2 Visualisierung der Daten

Jetzt wird in der Funktion `def read_table` die entstandene csv-Datei mit den Rohdaten eingelesen (Z. 606). Die dann folgenden Funktionen dienen in Zweiergruppen der Erstellung von drei Visualisierungen. Die jeweils erste Funktion nutzt den eingelesenen DataFrame, um Berechnungen auszuführen, welche in der zweiten Funktion zur Erstellung eines Diagramms verwendet werden. Für die Visualisierungen wird die Funktionsbibliothek `pygal` verwendet (siehe Mounier, k.D.). Da sich die Funktionen in ihrem Aufbau sehr ähnlich sind, wird im Folgenden nur auf zwei eingegangen.
`def calculations_for_lineplot` summiert in den Zeilen 266-274 jeweils pro Periode die Anzahl von genutzten männlichen, weiblichen und neutralen Substantiven auf und schreibt die Summe in eine neue Zeile. Nun kommen auch die Indexe „Beginn Periode X“ zum Einsatz, welche bei der Befüllung des Dictionarys eingefügt wurden: durch sie können die Tabellenzeilen, die zu einer Periode gehören, voneinander abgegrenzt werden. Aus den Summen werden die relativen Häufigkeiten berechnet und in eigene Zeilen im DataFrame geschrieben. Zuletzt wird der DataFrame auf die neu gebildeten Zeilen reduziert und als csv-Datei abgespeichert (Z. 275-296).
`def make_linepolot_words_used` nutzt die Daten der vorherigen Funktion für die Erstellung eines Liniendiagramms. Es werden Stil, Grafik- und Achsentitel festgelegt (Z. 301-307). Das Einspeisen der Daten in die Grafik erfolgt über das Auswählen von Zellen des DataFrames mithilfe der `.loc-Methode`. Das Liniendiagramm wird als svg-Datei bereitgestellt (Z. 308-319). Weitere Visualisierungen für die Verwendung der Substantive in Abhängigkeit zu Geschlecht und Parteizugehörigkeit werden in den Funktionen `def calculations_for_barchart_speakers`, `def make_barchart_speakers`, `def calculations_barchart_parties` und `def make_barchart_parties` erstellt (Z. 609-612).

5. Ergebnisse

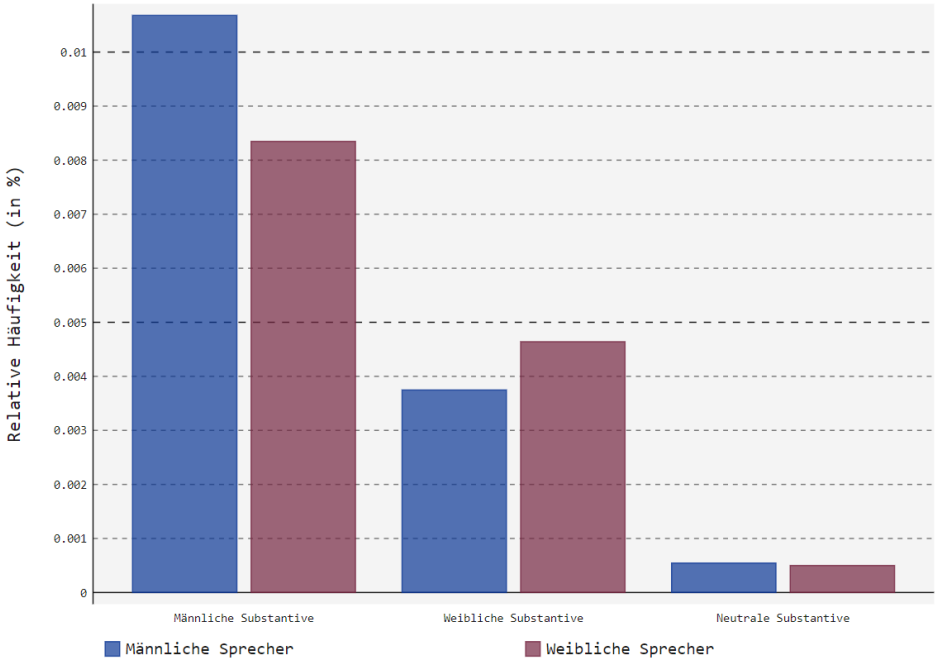
In diesem Kapitel wird ein Überblick über die Ergebnisse der automatisierten Analyse gegeben. Die niedrigen Ergebnis-Prozentzahlen sind darauf zurückzuführen, dass alle im Text enthaltenen Wörter gezählt und z.B. Stopwörter nicht im Voraus herausgefiltert wurden.
Abbildung 5.1 zeigt eine deutliche Veränderung der Nutzung männlicher und weiblicher Substantive über die Perioden hinweg (FF 1). Während die männlichen Substantive in Periode 1 bei 0,011% auf vergleichsweise hohem Niveau beginnen und dann in Periode 7 am häufigsten vorkommen (0,014%), fällt ihre Nutzung in Periode 14 stark ab und steigt in Periode 19 wieder etwas an auf 0,009%. Die Nutzung männlicher Begriffsformen hat sich somit gegenüber der ersten Wahlperiode leicht verringert. Mit 0,003% startet die Verwendung weiblicher Substantive bei einem deutlich niedrigeren Wert und erreicht einen Tiefpunkt in Periode 7 (0,002%). Danach ist ein starker Anstieg zu erkennen, der sich bis zur aktuellen Periode zieht und einen Endwert von fast 0,006% erreicht. Nur bei den neutralen Substantiven ist kaum eine Entwicklung zu erkennen. Sie starten bei weniger als 0,001%, was sich dann im Laufe der Perioden weiter verringert. Zusammenfassend können diese Daten die These von Stecker et al. (2021) bestätigen, dass in den aktuelleren Wahlperioden ein Anstieg in der Verwendung von weiblichen Substantiven zu vernehmen ist.

Abbildung 5.1: Nutzung der Substantive in verschiedenen Perioden



Auch der Zusammenhang zwischen Geschlecht der Abgeordneten und deren Verwendung verschiedener Substantive (FF 2) lässt sich anhand der erhobenen Daten nachvollziehen. Abbildung 5.2 zeigt zwar, dass Frauen mit einem Anteil von etwas über 0,008% häufiger männliche Substantive verwenden als weibliche (0,0046%). Dennoch werden weibliche Substantive häufiger von Frauen als von Männern verwendet. Diese Feststellung trifft umgekehrt ebenso auf männliche Sprecher zu, welche deutlich häufiger männliche Substantive verwenden (0,01%) als Frauen. Nur selten nutzen Männer weibliche Substantive in ihren Plenarreden (0,0037%) Bei der Verwendung von neutralen Substantiven ist kein nennenswerter Unterschied zwischen Männern und Frauen zu erkennen (beide 0,0005%).

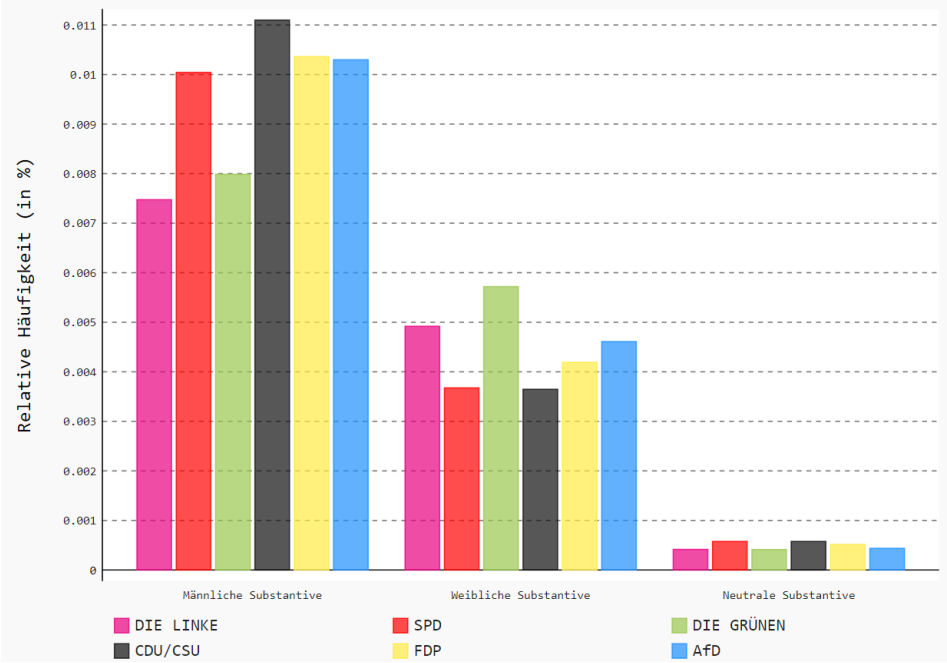
Abbildung 5.2: Nutzung von Substantiven in Abhängigkeit zum Geschlecht



Staffelt man die Verwendung der Substantive nach Parteizugehörigkeit (FF 3), so werden ebenfalls Abhängigkeiten deutlich. In Abbildung 5.3 werden zur Übersichtlichkeit nur die Parteien dargestellt, die in der jetzigen Wahlperiode im Bundestag vertreten sind. Zu erkennen ist, dass vor allem Mitglieder der CDU/CSU sehr häufig männliche Begriffsformen (0,011%) verwenden. Gleiches gilt für FDP, AfD und SPD (alle mit ca. 0,01%). Die Linke und Die Grünen liegen bei 0,0075% bzw. 0,008%. Bei der Verwendung weiblicher Substantive lässt sich der umgekehrte Trend feststellen: Die Grünen und Die Linke nutzen diese am häufigsten mit 0,0057% und ca. 0,005%, es folgt die AfD mit 0,0046%. CDU und SPD verwenden am seltensten weibliche Substantive (beide 0,0036%). Bei der Verwendung neutraler Begriffsformen ist erneut kaum ein Unterschied zu erkennen. Hier liegt der höchste Wert bei 0,0005% (SPD und CDU/CSU), der niedrigste bei 0,0004% (Die Grünen). Insgesamt bestätigen sich hier zu großen Teilen die Befunde von Stecker et al. (2021),

dass Parteien des linken Spektrums häufiger weibliche Substantive verwenden als konservative Parteien.

Abbildung 5.3: Nutzung von Substantiven in Abhängigkeit zur Parteizugehörigkeit



Wie die Visualisierungen zeigen, können die Befunde der Vergleichsstudie zu großen Teilen bestätigt werden, vereinzelt kommt es jedoch zu Abweichungen. Der vergleichsweise hohe Prozentwert für die Verwendung weiblicher Substantive bei der AfD könnte sich damit erklären lassen, dass die Partei erst seit der 19. Wahlperiode im Bundestag sitzt. So wurden deutlich weniger aber aktuellere Texte dieser Partei analysiert, was den Wert leicht verzerren könnte. Weitere Abweichungen von den Ergebnissen der Vergleichsstudie könnten dadurch erklärt werden, dass in der vorliegenden Ausarbeitung nur vier Wahlperioden analysiert wurden, während Stecker et al. (2021) alle Perioden betrachtet haben. Ein weiterer Grund für Abweichungen könnte die deutlich detailliertere und aufwendigere Vorarbeit der Forscher sein. Sie haben in einer Voranalyse häufige Substantive in ihrer männlichen und weiblichen Form aus den Plenarreden extrahiert und ihre Analyse darauf gestützt. In dieser Hausarbeit erfolgte die Erstellung der Abgleichslisten für die Substantive ohne Voranalyse, weshalb es möglich ist, dass die Listen nicht perfekt auf den analysierten Korpus abgestimmt sind. Zur Verwendung neutraler Begriffsformen kann auch in der vorliegenden Analyse nur schlecht eine Aussage getroffen werden, denn sie kommen in den Daten kaum vor.

6. Evaluation und Fazit

Die vorliegende Arbeit analysiert die Verwendung geschlechtergerechter Sprache im Deutschen Bundestag. Es werden zunächst die theoretischen Grundlagen zu diesem Thema eingeführt. Dann wird ein Algorithmus vorgestellt, der automatisiert die Verwendung männlicher, weiblicher und neutraler Begriffsformen in Abhängigkeit zu Wahlperiode, Geschlecht und Parteizugehörigkeit der Redenden im Bundestag erhebt. Die Ergebnisse der Analyse bestätigen in weiten Teilen die der Studie von Stecker et al. (2021) und zeigen, dass die Verwendung weiblicher Begriffsformen nach Wahlperiode 7 stark ansteigt. Außerdem ist erkennbar, dass Sprecherinnen häufiger weibliche Substantive verwenden als ihre männlichen Kollegen. Ein Parteizusammenhang zeigt sich ebenfalls: Sprechende aus konservativen Parteien verwenden gehäuft männliche Begriffsformen, während Die Linke und Die Grünen besonders häufig weibliche Substantive verwenden. Zu neutralen Begriffsformen kann keine Aussage getroffen werden, da sie sehr selten im Korpus vorkommen.
Im Hinblick auf Modularisierung und Effizienz des Programms gilt zu erwähnen, dass die einzelnen Sprechertexte zuverlässig erkannt und analysiert werden. Das Arbeiten mit verschiedenen Abgleichslisten für die Substantive, Parteien und Abgeordnete erscheint aufwendig, ist jedoch eine effiziente Methode, um nach bestimmten Strings zu suchen. Die Datenerhebung mit dem beschriebenen Algorithmus ist sehr transparent, da neben den Rohdaten auch die weiterverarbeiteten Daten ausgegeben werden. So wird die Nachnutzbarkeit und Nachvollziehbarkeit der Daten ermöglicht. Die Nutzung von teils zwei verschiedenen, aber ähnlichen Funktionen für die Perioden 1, 7 und 14, und 19 ist keine besonders kurze, aber dafür zuverlässige Methode. Aufgrund der sehr unterschiedlichen Detaillierungsgrade der XML-Auszeichnung wurden für die Datenerhebung deshalb meist zwei voneinander unabhängige, dafür aber kurze, modularisiert Funktionen erstellt. So kann ein hoher Modularisierungsgrad erreicht werden. Anzumerken ist jedoch, dass besonders die Funktionen, in denen Parteien gesucht oder die zugehörigen Daten weiterverarbeitet werden, ausladend sind. Dies liegt zum Einen an der Vielzahl der Parteien und deren Bezeichnungen, die in der Analyse von Bedeutung sind. Zum Anderen arbeitet die Kombination der Suchanfragen beispielsweise mit Booleschen Operatoren nicht zuverlässig.
An diese Ausarbeitung könnte eine Analyse aller Wahlerioden mit dem gleichen Algorithmus angeschlossen werden, um zu sehen, ob sich Abweichungen ergeben. Lohnenswert wäre auch der direkte Vergleich der Plenarreden zwischen verschiedenen deutschsprachigen Staaten oder zwischen den deutschen Landtagen. Hier könnten durchaus regionale Unterschiede zu erkennen sein. Zuletzt wäre sicherlich auch

die Verwendung verschiedener Begriffsformen im Zusammenhang mit dem Alter der Sprechenden interessant. Zu vermuten ist, dass Jüngere dem Gebrauch von weiblichen und neutralen Begriffsformen gegenüber offener sind als ältere Generationen. Insgesamt bietet das Thema der geschlechtergerechten Sprache noch viel Raum für tiefergehende Analysen. Gerade vor dem Hintergrund, dass geschlechtergerechte Sprache eine immer signifikantere Rolle im täglichen Sprachgebrauch einnehmen wird, ist hier ein großes Forschungspotenzial vorhanden.

Literaturvereichnis

Bundesministerium für Bildung, Wissenschaft und Forschung Österreich. (2018). *Geschlechtergerechte Sprache: Leitfaden im Wirkungsbereich des BMBWF* [18.08.2021].

Bundeszentrale für politische Bildung. (2017). *Frauenanteil im Deutschen Bundestag*. https://www.bpb.de/gesellschaft/gender/frauen-in-deutschland/49418/frauenanteil-im-deutschen-bundestag [23.08.2021].

Bundeszentrale für politische Bildung. (2020). *Bevölkerung nach Altersgruppen und Geschlecht*. https://www.bpb.de/nachschlagen/zahlen-und-fakten/soziale-situation-in-deutschland/61538/altersgruppen [12.08.2021].

Deutscher Bundestag. (k.D.). *Open Data*. https://www.bundestag.de/services/opendata [12.08.2021].

Deutscher Bundestag. (2015). *Bundestags-Plenar-Protokolle im XML-Format: Aufbau der Strukturdefinition - DTD*. https://www.bundestag.de/resource/blob/577234/f9159cee3e045cbc37dcd6de6322fcdd/dbtplenarprotokoll_kommentiert-data.pdf [23.08.2021].

Grüne Köln. *Die Frauenquote bei BÜNDNIS 90/ DIE GRÜNEN - ein Erfolgsmodell und wie es richtig funktioniert*. (k.D.) https://www.gruenekoeln.de/fileadmin/user_upload/Frauenstatut_-_ein_Erfolgsmodell_und_wie_es_funktioniert.pdf [13.08.2021].

Höhne, B. (2020). *Frauen in Parteien und Parlamenten. Innerparteiliche Hürden und Ansätze für Gleichstellungspolitik*. Bundeszentrale für politische Bildung. https://www.bpb.de/apuz/315247/frauen-in-parteien-und-parlamenten [16.08.2021].

Knoke, M. (2017). *Wie „Gender“darf die Sprache werden?* Goethe-Institut. https://www.goethe.de/ins/hu/de/kul/sup/klt/21458969.html [12.08.2021].

Kunze, C. (2015). *Geschlechtergerecht in Sprache und Bild: Ein Leitfaden.* http://www.geschkult.fu-berlin.de/service/frauenbeauftragte/ressourcen/leitfaden_gendergerechte_sprache-1.pdf [12.08.2021].

Maron, M., Schneider, W., Krämer, W. & Kraus, J. (2019). *Schluss mit Gender-Unfug!* Verein Deutscher Sprache. https://vds-ev.de/gegenwartsdeutsch/gendersprache/gendersprache-unterschriften/schluss-mit-dem-gender-unfug/ [16.08.2021].

Mounier, F. (k.D.). *Pygal Documentation.* http://www.pygal.org/en/stable/documentation/index.html [20.08.2021].

Pandas Development Team. (k.D.). *Pandas Documentation.* https://pandas.pydata.org/docs/index.html [20.08.2021].

Payr, F. (Hrsg.). (2021). *Von Menschen und Mensch*innen: 20 gute Gründe, mit dem Gendern aufzuhören.* Springer. https://doi.org/10.1007/978-3-658-33127-6

Payr, F. (2021). Wie sexistisch ist das Gendern? In F. Payr (Hrsg.). *Von Menschen und Mensch*innen: 20 gute Gründe, mit dem Gendern aufzuhören* (S. 59–67). Springer. https://doi.org/10.1007/978-3-658-33127-6_8

Pollatschek, N. (2020). *Gendern macht die Diskriminierung nur noch schlimmer.* https://www.tagesspiegel.de/kultur/deutschland-ist-besessen-von-genitalien-gendern-macht-die-diskriminierung-nur-noch-schlimmer/26140402.html [13.08.2021].

Rahlf, T. (Hrsg.). (2015). *Deutschland in Daten. Zeitreihen zur Historischen Statistik.* Bundeszentrale für politische Bildung. https://www.econstor.eu/bitstream/10419/124185/1/4938_zb_dtindaten_150714_online.pdf [16.08.2021].

Richardson, L. (k.D.). *Beautiful Soup Documentation.* https://www.crummy.com/software/BeautifulSoup/bs4/doc/ [20.08.2021].

Stecker, C., Müller, J., Blätte, A. & Leonhardt, C. (2021). *The evolution of gender-inclusive language. Evidence from the German Bundestag, 1949-2021.* https://doi.org/10.31219/osf.io/fcsmz

Wesian, J. (2007). *Sprache und Geschlecht: Eine empirische Untersuchung zur "geschlechtergerechten Sprache"* [SASI Studentische Arbeitspapiere zu Sprache und Interaktion]. Westfälische Wilhelms-Universität Münster, Münster. http://arbeitspapiere.sprache-interaktion.de/stud/arbeitspapiere/arbeitspapier13.pdf [16.08.2021].

Wikipedia. (2021). *Frauenanteil im Deutschen Bundestag seit 1949*. https://de.wikipedia.org/wiki/Frauenanteil_im_Deutschen_Bundestag_seit_1949 [13.08.2021].

Wikipedia. (2021). *Geschlechtergerechte Sprache*. https://de.wikipedia.org/wiki/Geschlechtergerechte_Sprache [12.08.2021].

Anhang

A. Vergleichslisten

```
1 Katrin , Albsteiger , Agnes , Alpers , Luise , Amtsberg , Kerstin , Andreae ,
    Niels , Annen , Ingrid , Arndt-Brauer , Heike , Baehrens , Annalena ,
    Baerbock , Ulrike , Bahr , Bettina , Bähr-Losse , Dorothee , Bär , Katarina
    , Barley , Doris , Barnett , Bärbel , Bas , Sabine , Bätzing-Lichtenthäler
    , Marieluise , Beck , Veronika , Bellmann , Sybille , Benning , Andr  ,
    Berghegger , Ute , Bertram , Karin , Binder , Heidrun , Bluhm , Maria , Bö
    hmer , Franziska , Brantner , Heike , Brehmer , Agnieszka , Brugger ,
    Christine , Buchholz , Eva , Bulling-Schröter , Edelgard , Bulmahn ,
    Gitta , Connemann , Petra , Crone , Daniela , DeRidder , Alexandra , Dinges
    -Dierig , Sabine , Dittmar , Katja , Dörner , Marie-Luise , Dött , Elvira ,
    Drobinski-Weiß , Katharina , Dröge , Iris , Eberl , Annette , Margaret ,
    Pantel , Poschmann , Jutta , Eckenbach , Michaela , Engelmeier-Heite ,
    Petra , Ernstberger , Saskia , Esken , Karin , Evers-Meyer , Elke , Ferner ,
    Ute , Finckh-Krämer , Ingrid , Fischbach , Maria , Flachsbarth , Gabriele
    , Fograscher , Dagmar , Freitag , Astrid , Freudenstein , Iris , Gleicke ,
    Angelika , Glöckner , Nicole , Gohlke , Diana , Golze , Katrin , Göring-
    Eckardt , Ulrike , Gottschalck , Kerstin , Griese , Ursula , Groden-
    Kranich , Gabriele , Groneberg , Astrid , Grotelüschen , Groth , Monika ,
    Grütters , Herlind , Gundelach , Bettina , Hagedorn , Rita , Hagl-Kehl ,
    Anja , Hajduk , Heike , Hänsel , Britta , Haßelmann , Mechthild , Heil ,
    Rosemarie , Hein , Gabriela , Heinrich , Uda , Heller , Barbara , Hendricks
    , Heidtrud , Henn , Marion , Herdan , Gabriele , Hiller-Ohm , Petra , Hinz ,
    Priska , Hinz , Inge , Höger , Eva , Högl , Bärbel , Höhn , Horb , Bettina ,
    Hornhues , Anette , Hübinger , Sigrid , Hupach , Christina , Jantz , Ulla ,
    Jelpke , Sylvia , Jörrißen , Christina , Kampmann , Susanna , Karawanskij
    , Anja , Karliczek , Kerstin , Kassner , Gabriele , Katzmarek , Ronja ,
    Kemmer , Marina , Kermer , Katja , Keul , Katja , Kipping , Maria , Klein-
    Schmeink , Bärbel , Kofler , Daniela , Kolbe , Birgit , Kömpel , Sylvia ,
    Kotting-Uhl , Kordula , Kovac , Anette , Kramme , Jutta , Krellmann ,
    Angelika , Krüger-Leißner , Bettina , Kudla , Helga , Kühn-Mengel ,
    Renate , Künast , Katrin , Kunert , Christine , Lambrecht , Katharina ,
    Landgraf , Barbara , Lanzinger , Silke , Launert , Monika , Lazar , Sabine ,
    Leidig , Katja , Leikert , Steffi , Lemke , Ursula , vonderLeyen , Antje ,
```

Lezius , Andrea , Lindholz , Patricia , Lips , Gabriele ,Lösekrug–Möller , Hiltrud , Lotze , Gesine ,Lötzsch , Claudia ,Lücking–Michel , Daniela , Ludwig , Kirsten ,Lühmann , Karin , Maag , Yvonne , Magwas , Nicole , Maisch , Birgit , Malecha–Nissen , Gisela , Manderla , Katja , Mast , Hilde , Mattheis , Birgit ,Menz , Angela , Merkel , Maria , Michalk , Irene , Mihalic , Susanne , Mittag , Cornelia ,Möhring , Marlene , Mortler , Elisabeth , Motschmann , Niema , Movassat , Bettina ,Müller , Beate ,Müller –Gemmeke , Michelle ,Müntefering , Andrea , Nahles , Michaela , Noll , Julia , Obermeier , Friedrich , Ostendorff , Ingrid , Pahlmann , Sylvia , Petra , Pau , Elisabeth , Paus , Sibylle , Pfeiffer , Jeannine , Pflugradt , Sabine , Brigitte , Pothmer , Simone , Raatz , Kerstin , Radomski , Mechthild , Rawert , Katherina , Reiche , Tabea , Nina , Elfi , Schulz–Asche , Rita , Kirsten , Funcke , Timm, Heinrich , Carola , Reimann , Martina , Renner , Iris , Ripsam , Petra , Rode–Bosse , Kathrin ,Rösel ,Rößner , Claudia , Roth , Corinna ,Rüffer , Susann ,Rüthrich , Sarah , Ryglewski , Annette , Sawade , Anita , Schäfer , Elisabeth , Scharfenberg , Ursula , Schauws , Annette , Schavan , Scheer , Marianne , Schieder , Jana , Schimke , Dorothee , Schlegel , Dagmar , Gabriele , Ulla , Scho–Antwerpes , Nadine , Schön , Kristina , Schröder , Ursula , Schulte , Kordula , Rita , Schwarzelühr–Sutter , Christina , Schwarzer , Petra , Sitte , Svenja , Stadler , Martina ,Stamm–Fibich , Carola , Stauche , Sonja , Steffen , Erika , Steinbach , Kersten , Steinke , Stockhofe , Karin , Strenz , Lena , Strothmann , Sabine ,Sütterlin –Waack , Kerstin , Tack , Tackmann , Claudia , Tausend , Karin , Thissen , Antje , Tillmann , Astrid , Timmermann–Fechter , Julia , Verlinden , Kathrin , Vogler , Ute , Vogt , Christel ,Voßbeck–Kayser , Sahra , Wagenknecht , Weber , Widmann–Mauz , Bergmann–Pohl , Buntenbach , Renate , Maria , Doris , Wagner , Beate , Walter–Rosenheimer , Nina , Warken , Halina , Wawzyniak , Gabi , Anja , Weisgerber , Sabine , Weiss , Katrin , Werner , Andrea , Wicklein , Annette , Valerie , Wilms , Elisabeth , Winkelmeier–Becker , Dagmar ,Wöhrl , Waltraud , Wolff , Birgit ,Wöllert , Barbara , Woltmann ,Emmi, Zeulner , Dagmar , Ziegler , Pia , Zimmermann , Sabine , Zimmermann , Gudrun , Zollner , Brigitte , Zypries , Brigitte , Adler , Ilse , Aigner , Ina , Albowitz , Gila , Altmann , Ingrid , Arndt–Brauer , Monika , Balt , Doris , Barnett , Brigitte , Baumeister , Marieluise , Beck , Ingrid , Becker–Inglau , Angelika , Beer , Sabine , Merkel , Grietje , Bettin , Petra , Bierwirth , Renate , Blank , Petra , Bläss , Antje , Blumenthal , Maria ,Böhmer , Sylvia , Bonitz , Maritta ,Böttcher , Anni , Brandt–Elsweier , Hildebrecht , Braun , Monika , Brudlewsky , Eva , Bulling–Schröter , Edelgard , Bulmahn , Annelie , Verena , Ulla , Burchardt , Marion , Caspers–Merk , Herta ,Däubler–Gmelin , Christel , Deichmann , Diemers , Amke, Dietert–Scheuer , Marie–Luise ,Dött , Thea ,Dückert , Heidemarie ,

```
Ehlert , Eichhorn , Franziska , Eichstädt–Bohlig , Uschi , Eid , Marga ,
Elser , Petra , Ernstberger , Anke , Eymer , Ilse , Falk , Annette , Faße ,
Ingrid , Fischbach , Andrea , Fischer , Ulrike , Flach , Gabriele ,
Fograscher , Iris , Follak , Dagmar , Freitag , Gisela , Frick , Lilo ,
Friedrich , Anke , Fuchs , Ruth , Fuchs , Monika , Ganseforth , Michaela ,
Geiger , Iris , Gleicke , Katrin , Göring–Eckardt , Renate , Gradistanac ,
Angelika , Graf , Monika , Griefahn , Kerstin , Griese , Rita , Grießhaber ,
Bärbel , Grygier , Christel , Hanewinckel , Anke , Hartnagel , Hermenau ,
Ulla , Evelyn , Gudrun , Nicolette , Lemke , Ursula , Marquardt , Ute , Gerda
, Hasselfeldt , Nina , Hauer , Ursula , Heinen–Esser , Barbara , Hendricks
, Antje , Monika , Heubaum , Kristin , Heyne , Jelena , Hoffmann , Iris ,
Hoffmann , Ulrike , Höfken , Barbara , Höll , Ingrid , Holzhüter , Birgit ,
Homburger , Christel , Humme , Barbara , Imhof , Brunhilde , Irber ,
Gabriele , Iwersen , Susanne , Jaffke , Renate , Jäger , Ilse , Janz ,
Jelpke , Sabine , Jünger , Irmgard , Karwatzki , Sabine , Kaspereit , Susanne ,
Kastner ,
Kenzler , Marianne , Klappert , Siegrun , Klemmer , Heidi , Knake–Werner ,
Monika , Knoche ,
Kopp , Eva–Maria , Kors , Karin , Kortmann , Angelika , Köster–Loßack , Anette
, Kramme ,
Kressl , Martina , Krogmann , Angelika , Krüger–Leißner , Helga , Kühn–
Mengel , Ute , Kumpf ,
Christine , Lambrecht , Brigitte , Lange , Christine , Lehder , Waltraud ,
Lehn , Steffi ,
Vera , Lengsfeld , Ina , Lenke , Elke , Leonhard , Sabine , Leutheusser–
Schnarrenberger ,
Lietz , Heidi , Lippmann , Christa , Lörcher , Gabriele , Lösekrug–Möller ,
Erika , Lotz ,
Lötzer , Christine , Lucyga , Christa , Luft , Heidemarie , Lüth , Pia , Maier ,
Angela ,
Ulrike , Mascher , Ingrid , Matthäus–Maier , Heide , Mattischeck , Ulrike ,
Mehl , Angela ,
Ulrike , Merten , Angelika , Mertens , Ursula , Mogg , Jutta , Müller , Kerstin ,
Müller ,
Andrea , Nahles , Kerstin , Naumann , Rosel , Neuhäuser , Christa , Nickels ,
Edith , Niehuis ,
Claudia , Nolte , Leyla , Onur , Christine , Ostrowski , Petra , Pau , Beatrix ,
Philipp ,
Cornelia , Pieper , Marlies , Pretzlaff , Simone , Probst , Karin , Rehbock–
Zureich , Christa ,
Reichard , Carola , Reimann , Margot , von , Renesse , Renate , Rennebach ,
Hannelore , Rönsch ,
```

Gudrun , Roos , Birgit , Roth , Marlene , Rupprecht , Anita , Schäfer , Gudrun , Schaich–Walch ,
Christine , Scheel , Christina , Schenk , Irmingard , Schewe–Gerigk , Dagmar , Schmidt , Silvia ,
Schmidt , Ulla , Schmidt , Regina , Schmidt–Zadel , Birgit , Schnieber–Jastram , Andreas ,
Schockenhoff , Gisela , Hilbrecht , Erika , Schuchardt , Brigitte , Schulte , Ilse , Schumann ,
Irmgard , Schwaetzer , Angelica , Schwall–Düren , Marita , Sehn , Marion , Seib , Gudrun ,
Serowiecki , Erika , Simm , Sigrid , Skarpelis–Sperk , Cornelie , Sonntag–Wolgast , Bärbel ,
Sothmann , Margarete , Späte , Margrit , Spielmann , Antje–Marie , Stehen , Erika , Steinbach ,
Dorothea , Störr–Ritter , Rita , Streb–Hesse , Rita , Süssmuth , Jella , Teuchner , Susanne ,
Tiemann , Uta , Titze–Stecher , Edeltraut , Töpfer , Adelheid , Tröscher , Simone , Violka ,
Vogt , Antje , Vollmer , Angelika , Volquartz , Sylvia , Voß , Andrea , Voßhoff , Konstanze ,
Wegner , Hildegard , Wester , Lydia , Westrich , Inge , Wettig–Danielmeier , Margrit , Wetzel ,
Annette , Widmann–Mauz , Heidemarie , Wieczorek–Zeul , Brigitte , Wimmer , Barbara , Wittig ,
Wohlleben , Dagmar , Wöhrl , Hanna , Wolf , Margareta , Wolf , Waltraud , Wolff , Heidemarie ,
Wright , Elke , Wülfing , Uta , Zapf , Ursula , Benedix , Lieselotte , Berger , Lenelotte ,
von , Bothmer , Herta , Däubler–Gmelin , Elfriede , Eilers , Katharina , Focke , Liselotte ,
Erna–Maria , Geier , Angela , Grützmann , Ingeborg , Häckel , Antje , Huber , Agnes , Hürland ,
Renate , Lepsius , Barbara , Lüdemann , Hanna , Neumeister , Elisabeth , Orth , Doris , Pack ,
Liselotte , Pieser , Wiltrud , Rehlen , Annemarie , Renger , Paula , Riede , Marie , Schlei ,
Ursula , Schleicher , Helga , Schuchardt , Waltraud , Steinhauer , Maria , Stommel , Helga ,
Irma , Tübler , Roswitha , Verhülsdonk , Hanna , Walz , Helga , Wex , Luise , Albertz , Lisa ,
Albrecht , Maria , Ansorge , Thea , Arnold , Anna , Maria , Bieganowski , Else , Brökelschen ,

```
37 Maria , Dietz , Clara ,Döhring , Margarete , Gröwel , Margarete ,Hütter ,
       Herta , Ilk ,
38 Imig , Elfriede , Jaeger , Margot , Kalinke , Irma , Keilhack , Liesel , Kipp-
       Kaule , Lisa ,
39 Korspeter , Anni , Krahnstöver , Agnes , Katharina , Maxsein , Friederike ,
       Nadig , Maria ,
40 Niggemeyer , Maria , Probst , Luise , Rehling , Julie ,Rösch , Marta ,
       Schanzenbach ,Käte ,
41 Strobel , Gertrud , Strohbach , Helene , Weber , Jeanette , Wolff , Helene ,
       Wessel
```

Listing 1: Liste weiblicher Abgeordneten

```
1  Abgeordneter , Minister , Staatssekretär , Staatssekretäre ,
       Vorsitzender , Generalsekretär ,
2  Präsident , Präsidenten , Politiker , Sportler , Lehrer , Bürger , Kollege ,
       Kollegen ,
3  Arbeitnehmer , Soldat , Soldaten , Bundesminister , Vertreter ,
       Verbraucher , Studenten ,
4  Mitarbeiter , Sozialdemokrat , Sozialdemokraten , Partner , Freund ,
       Freunde , Beamte , Beamter ,
5  Beamten , Finanzminister , Arzt , Ärzte , Professor , Professoren ,
       Professors , Bayern , Bayer ,
6  Rentner , Arbeitgeber , Ministerpräsident , Ministerpräsidenten ,
       Gesetzgeber , Außenminister ,
7  Bauer , Unternehmer , Patient , Patienten , Demokraten , Demokrat , Bänker ,
       Student , Anwalt ,
8  Schüler , Polizist , Polizisten , Arbeiter , Erzieher , Altenpfleger ,
       Assistent , Assistenten ,
9  Vorstand , Ingenieur , Ingenieure , Betriebswirt , Betriebswirte ,
       Informatiker , Bäcker ,
10 Dolmetscher , Friseur , Verkäufer , Flugbegleiter , Pilot , Piloten , Jurist
       , Juristen ,
11 Anwälte , Landwirt , Landwirte , Handwerker , Winzer , Techniker ,
       Vorsitzender , Parlamentarier ,
12 Journalist , Journalisten , Bundestagspräsident , Bundestagsprä
       sidenten , Vizepräsident ,
13 Vizepräsidenten , Alterspräsident , Alterspräsidenten , Kandidat ,
       Kandidaten , Kanzlerkandidat ,
14 Kanzlerkandidatinnen , Geschäftsführer , Stellvertreter , Wähler ,
       Demonstrant , Demonstranten ,
15 Protestant , Protestanten , Anleger , Analyst , Analysten , Benutzer ,
       Berater ,
16 Berichterstatter , Betreuer , Patient , Patienten , Helfer , Herren , Vater ,
       Väter
```

Listing 2: Liste männlicher Begriffsformen

```
1  Kollegin , Kolleginnen , Ministerin , Ministerinnen , Staatssekretärin ,
2  Staatssekretärinnen , Vorsitzende , Generalsekretärin , Generalsekretä
       rinnen , Präsidentin ,
3  Präsidentinnen , Politikerin , Politikerinnen , Sportlerin ,
       Sportlerinnen ,
4  Lehrerin , Lehrerinnen , Bürger , Bürgerinnen , Kollegin , Kolleginnen ,
       Bundeskanzlerinnen ,
5  Arbeitnehmerin , Arbeitnehmerinnen , Soldatin , Soldatinnen , Damen,
       Mutter , Mütter
6  Bundesministerin , Bundesministerinnen , Vertreterin , Vertreterinnen ,
       Verbraucherin ,
7  Verbraucherinnen , Mitarbeiterin , Mitarbeiterinnen , Sozialdemokratin
       , Sozialdemokratinnen ,
8  Partnerin , Partnerinnen , Freundin , Freundinnen , Beamtin , Beamtinnen ,
       Finanzministerin ,
9  Ärztin , Ärztinnen , Professorinnen , Professorin , Bayerin , Bayerinnen ,
       Rentnerinnen ,
10 Rentnerin , Arbeitgeberin , Arbeitgeberinnen , Ministerpräsidentin ,
       Ministerpräsidentinnen ,
11 Gesetzgeberin , Gesetzgeberinnen , Außenministerin , Auß
       enministerinnen , Bäuerin , Bäuerinnen ,
12 Unternehmerin , Unternehmerinnen , Patientin , Patientinnen , Demokratin
       , Demokratinnen ,
13 Bankerin , Bankerinnen , Studentin , Studentinnen , Schülerinnen , Schü
       lerin , Polizistin ,
14 Polizistinnen , Arbeiterin , Arbeiterinnen , Erzieherinnen , Erzieherin ,
       Altenpflegerin ,
15 Altenpflegerinnen , Assistentin , Assistentinnen , Vorständin , Vorstä
       ndinnen , Ingenieurin ,
16 Ingenieurinnen , Betriebswirtin , Betriebswirtinnen , Informatikerin ,
       Informatikerinnen ,
17 Bäckerin , Bäckerinnen , Dolmetscherin , Dolmetscherinnen , Friseurin ,
       Friseurinnen , Friseuse ,
18 Verkäuferin , Verkäuferinnen , Flugbegleiterin , Flugbegleiterinnen ,
       Pilotin , Pilotinnen ,
19 Juristin , Juristinnen , Anwältin , Anwältinnen , Landwirtinnen ,
       Landwirtin , Handwerkerin ,
20 Handwerkerinnen , Winzerin , Winzerinnen , Technikerin , Technikerinnen ,
       Vorsitzende ,
21 Parlamentarierin , Journalistin , Journalistinnen , Bundestagsprä
       sidentin ,
```

```
22 Bundestagspräsidentinnen , Vizepräsidentin , Vizepräsidentinnen ,
       Alterspräsidentin ,
23 Alterspräsidentinnen , Kandidatin , Kandidatinnen , Kanzlerkandidatin ,
       Kanzlerkandidatinnen ,
24 Geschäftsführerin , Geschäftsführerinnen , Stellvertreterin ,
       Stellvertreterinnen ,Wählerin ,
25 Wählerinnen , Demonstrantin , Demonstrantinnen , Protestantin ,
       Protestantinnen , Anlegerin ,
26 Anlegerinnen , Beobachterin , Beobachterinnen , Analystin , Analystinnen
       , Benutzerin ,
27 Benutzerinnen , Beraterin , Beraterinnen , Berichterstatterin ,
       Berichterstatterinnen ,
28 Betreuerin , Betreuerinnen , Patientin , Patientinnen , Helferin ,
       Helferinnen ,Dame
```

Listing 3: Liste weiblicher Begriffsformen

```
1  Studierende , Studierenden , Lehrende , Lehrenden , Lehrkräfte , Lehrkraft
       , Lehrperson ,
2  Mitarbeitende , Mitarbeitenden , Führungskraft , Führungskräfte ,
       Arbeitskraft , Arbeitskräfte ,
3  Teilnehmende , Teilnehmenden , Vorsitzenden , Angestellten , Pflegende ,
       Pflegenden ,
4  Gutachtende , Gutachtenden , Nutzende , Nutzenden , Dozierende ,
       Dozierenden , Helfende ,
5  Helfenden , Unterstützende , Unterstützenden , Interessierte ,
       Interessierten , Promovierende ,
6  Promovierende , Anwesende , Anwesenden , Verantwortlichen ,
       Verantwortliche , Referierende ,
7  Vorsitz , Geschäftsführung , Anwesende , Anwesenden , Erwerbslosen , Angeh
       örigen ,
8  Steuerpflichtigen , Leitung , Leitende , Helfende , Helfenden , Leitenden ,
       Fachkraft ,
9  Fachkräfte , Teilzeitkraft , Teilzeitkräfte , Ansprechperson ,
       Ansprechpersonen ,
10 Wahlberechtigte , Wahlberechtigen , Zuhörende , Zuhörenden , Beteiligte ,
       Beteiligten ,
11 Auftraggebende , Auftraggebenden , Interessierte , Interessierten ,
       Mitglied , Mitglieder ,
12 Mitgliedern , Beschäftige , Beschäftigen , Senatsmitglieder ,
       Senatsmitgliedern ,
13 Parlamentsmitglieder , Parlamentsmitgliedern , Wahlleitung , Präsidium
       , Reinigungskraft ,
14 Regierende , Regierenden , Demonstrierende , Demonstrierenden ,
       Protestierende ,
15 Protestierenden , Antragstellende , Antragstellenden , Abnehmende ,
       Abnehmenden ,
16 Bezugsperson , Alumni , Abteilungsleitung , Abteilungsleitende ,
       Administration , Admin ,
17 Administrierende , Administrierenden , Studierte , Studierten ,
       Agierende , Agierenden ,
18 Handelnde , Handelnden , Aktive , Aktiven , Anteilshabende ,
       Anteilshabenden , Alkoholsüchtige ,
19 Alkoholsüchtigen , Allergiegeplagte , Allergiegeplagten , Alltagshilfe
       , Pflegefachkraft ,
20 Pflegekraft , Ehemalige , Ehemaligen , Anfangende , Anfangenden ,
       Beginnende , Beginnenden ,
```

```
21 Analysierende , Analysierenden , Unerfahrene , Unerfahrenen , Mitglied ,
       Mitglieder , Angelnde ,
22 Angelnden , Angreifende , Angreifenden , Gefolgschaft , Animierende ,
       Animierenden ,
23 Anlegenden , Betreibende , Betreibenden , Vorgesetzen , Helfende ,
       Helfenden , Eltern
24 Anlegende , Rechtsvertretung , Arbeitgebende , Arbeitgebenden ,
       Teammitglieder , Assistenz ,
25 Auftraggebende , Auftraggebenden , Auftragnehmende , Auftragnehmenden ,
       Beobachtende ,
26 Beobachtenden , Auszubildende , Auszubildenden , Servicekraft ,
       Servicekräfte , Bedienstete ,
27 Bediensteten , Berichterstattende , Berichterstattenden ,
       Staatsoberhaupt , Vorgesetzte
```

Listing 4: Liste neutraler Begriffsformen

```
1 (CDU):,(CSU):,(CDU und CSU):,(CDU/CSU):,(BP):,(SPD):,(BÜNDNIS
      90/DIE GRÜNEN):,GRÜNEN):,
2 (DIE LINKE):,LINKE):,(FDP):,(F D P ):,(KPD):,(WAV):,(DP):,(FU)
      :,(FRAKTIONSLOS):,(PDS):,
3 (Z):,(DRP):,(AfD):
```

Listing 5: Liste für Parteiabfrage (Perioden 1-14)

```
1 CDU,CSU,CDU und CSU,CDU/CSU,BP,SPD,BÜNDNIS,DIE,FDP,F D P ,KPD,
2 WAV,DP,FU,FRAKTIONSLOS,PDS,Z,DRP,AfD
```

Listing 6: Liste für Parteiabfrage mit Beautiful Soup (Periode 19)

B. Programmcode

Der vollständige Programmcode ist ebenfalls digital abrufbar unter `https://github.com/s2lascmi/Hausarbeit_ProgrammierenI`.

```
import glob
from bs4 import BeautifulSoup
import re
import pandas as pd
from os.path import basename
import numpy as np
import pygal
from pygal.style import CleanStyle
from pygal.style import Style

# eigene Farbwerte entsprechend der Parteifarben definieren
# rosa       rot        grün       schwarz      gelb       hellblau
custom_style = Style(colors=('#E80080', '#FF0000', '#9BC850', '#0F0F0F', '#ffeb44', '#1E90FF',))


# bereits erstellte txt-Listen einlesen und leicht bereinigen
def read_list(filename):
    with open(filename, "r", encoding="utf8") as infile:
        list_import = infile.read()
        list_import = re.sub(r"\n", " ", list_import, flags=re.S)
        list_import = re.sub(", ", ",", list_import, flags=re.S)
        list_import = list_import.split(",")
        return list_import


# XML-Plenarprotokolle einlesen
def read_textfile(textfile):
    with open(textfile, "r", encoding="utf8") as infile:
        text = infile.read()
```

```
30          return text
31
32
33  # XML-Dokumente bereinigen
34  def clean_texts(text):
35      cleansed = re.sub(r"\t", " ", text)
36      cleansed = re.sub("\.", "", cleansed)
37      cleansed = re.sub(",", "", cleansed)
38      cleansed = re.sub("\?", "", cleansed)
39      cleansed = re.sub("!", "", cleansed)
40      cleansed = re.sub("\) :", "):", cleansed)
41      # Flag re.S sucht über mehrere Zeilen mit "." als special
        character
42      # für Texte der ersten Periode
43      cleansed = re.sub(r"<\?xml(.*?) eröffnet.", " ", cleansed,
        flags=re.S)
44      cleansed = re.sub(r"\(Schluß der Sitzung:(.*?)</DOKUMENT>",
        "", cleansed, flags=re.S)
45      cleansed = re.sub(r"\(Schluß: (.*?)</DOKUMENT>", "",
        cleansed, flags=re.S)
46      cleansed = re.sub(r"\(Schluss: (.*?)</DOKUMENT>", "",
        cleansed, flags=re.S)
47      # für Texte der 7. und 14. Periode
48      cleansed = re.sub(r"<?xml(.*?)Beginn:", "", cleansed, flags=
        re.S)
49      # für Texte der 19. Periode
50      cleansed = re.sub(r"<\?xml (.*?) <rede id", "<rede id",
        cleansed, flags=re.S)
51      cleansed = re.sub(r"</sitzungsverlauf(.*?)</
        dbtplenarprotokoll>", "", cleansed, flags=re.S)
52      cleansed = re.sub(r"<kommentar>(.*?)</kommentar>", "",
        cleansed, flags=re.S)
53      cleansed = re.sub(r"<", " <", cleansed, flags=re.S)
54      cleansed = re.sub(r"</", " </", cleansed, flags=re.S)
55      cleansed = re.sub(r">", "> ", cleansed, flags=re.S)
56      return cleansed
57
58
59  # sucht alle Sprecher anhand der Angabe der Partei in Klammern
        und mit : raus, da diese Kombination nur
60  # vorkommt, wenn ein neuer Sprecher beginnt
```

```
61 # mehrere Suchbefehle für eine Partei, da die Parteien in den
       verschiedenen Perioden unterschiedlich bezeichnet werden
62 def split_text_new_speaker(cleansed):
63     list_search_results = []
64     find_all_speakers_CDU1 = re.finditer("\(CDU\):", cleansed)
65     for item in find_all_speakers_CDU1:
66         list_search_results.append(item.end())
67     find_all_speakers_CDU2 = re.finditer("\(CSU\):", cleansed)
68     for item in find_all_speakers_CDU2:
69         list_search_results.append(item.end())
70     find_all_speakers_CDU3 = re.finditer("\(CDU und CSU\):",
       cleansed)
71     for item in find_all_speakers_CDU3:
72         list_search_results.append(item.end())
73     find_all_speakers_CDU4 = re.finditer("\(CDU/CSU\):",
       cleansed)
74     for item in find_all_speakers_CDU4:
75         list_search_results.append(item.end())
76     find_all_speakers_CDU5 = re.finditer("\(BP\):", cleansed)
77     for item in find_all_speakers_CDU5:
78         list_search_results.append(item.end())
79     find_all_speakers_CDU6 = re.finditer("\(CDU CSU\):",
       cleansed)
80     for item in find_all_speakers_CDU6:
81         list_search_results.append(item.end())
82     find_all_speakers_SPD = re.finditer("\(SPD\):", cleansed)
83     for item in find_all_speakers_SPD:
84         list_search_results.append(item.end())
85     find_all_speakers_GRUENE = re.finditer("\(BÜNDNIS 90/DIE GRÜ
       NEN\):", cleansed)
86     for item in find_all_speakers_GRUENE:
87         list_search_results.append(item.end())
88     find_all_speakers_LINKE1 = re.finditer("\(DIE LINKE\):",
       cleansed)
89     for item in find_all_speakers_LINKE1:
90         list_search_results.append(item.end())
91     find_all_speakers_LINKE2 = re.finditer("\(PDS\):", cleansed)
92     for item in find_all_speakers_LINKE2:
93         list_search_results.append(item.end())
94     find_all_speakers_FDP1 = re.finditer("\(F D P \):", cleansed
       )
95     for item in find_all_speakers_FDP1:
```

```
96          list_search_results.append(item.end())
97      find_all_speakers_FDP2 = re.finditer("\(FDP\):", cleansed)
98      for item in find_all_speakers_FDP2:
99          list_search_results.append(item.end())
100     find_all_speakers_KPD = re.finditer("\(KPD\):", cleansed)
101     for item in find_all_speakers_KPD:
102         list_search_results.append(item.end())
103     find_all_speakers_WAV = re.finditer("\(WAV\):", cleansed)
104     for item in find_all_speakers_WAV:
105         list_search_results.append(item.end())
106     find_all_speakers_DP = re.finditer("\(DP\):", cleansed)
107     for item in find_all_speakers_DP:
108         list_search_results.append(item.end())
109     find_all_speakers_FU = re.finditer("\(FU\):", cleansed)
110     for item in find_all_speakers_FU:
111         list_search_results.append(item.end())
112     find_all_speakers_FRAKTIONSLOS = re.finditer("\(Fraktionslos
    \):", cleansed)
113     for item in find_all_speakers_FRAKTIONSLOS:
114         list_search_results.append(item.end())
115     find_all_speakers_PDS = re.finditer("\(PDS\):", cleansed)
116     for item in find_all_speakers_PDS:
117         list_search_results.append(item.end())
118     find_all_speakers_Zentrum = re.finditer("\(Z\):", cleansed)
119     for item in find_all_speakers_Zentrum:
120         list_search_results.append(item.end())
121     find_all_speakers_DRP = re.finditer("\(DRP\):", cleansed)
122     for item in find_all_speakers_DRP:
123         list_search_results.append(item.end())
124     list_search_results.append(len(cleansed))
125     list_search_results.sort()
126     return list_search_results
127
128
129 # findet den Start der Zeile, gleichbedeutend mit Beginn des
    Namens
130 # initial ist beginning_of_line das Ende der Parteiklammer, bspw
    . bei (SPD) die Klammer zu ")"
131 def find_start_of_line(cleaned_text, list_search_results, i):
132     beginning_of_line = list_search_results[i]
133     # checken, dass Index in cleaned text ist, damit Index nicht
    out of range
```

```
134     if len(cleaned_text) == beginning_of_line:
135         return
136     begin_found = False
137     while not begin_found:
138         str = cleaned_text[beginning_of_line]
139         if str == "\n":
140             begin_found = True
141         else:
142             beginning_of_line = beginning_of_line - 1
143             # nur 1 zurückgehen, weil '\n' als 1 gezählt wird
144     # beginning_of_line + 1 um nicht "\n" mitzunehmen
145     name_and_party = cleaned_text[int(beginning_of_line + 1):(
    int(list_search_results[i]))]
146     whole_speaker_text = cleaned_text[int(beginning_of_line + 1)
    :(int(list_search_results[i + 1]))]
147     return [name_and_party, whole_speaker_text]
148
149
150 # Erstellung des Dictionarys, in dem Geschlecht und Partei der
    Sprechenden als auch deren Verwendung der
151 # Substantive und Gesamtlänge des Sprechertextes notiert wird
152 def create_dictionary(parties, female_MPs, male_words,
    female_words, neutral_words, single_speakers_texts):
153     dict_words = {"Gender of speaker:": " ", "Party of speaker:"
    : " ", "Male words:": 0, "Female words:": 0,
154                   "Neutral words:": 0, "Total words:": 0}
155     word_tokens_name_party = single_speakers_texts[0].split()
156     # fehlerhafte Formatierungen umgehen mit return-Statement:
157     # für Sprechertexte, für die aufgrund fehlerhafter
    Formatierungen Partei und Name nicht richtig
158     # ermittelt werden können, wird kein Dictionary erstellt
159     if len(word_tokens_name_party) < 2:
160         return
161     # nach Partei suchen und ins Dictionary eintragen
162     party_name = word_tokens_name_party[-1]
163     if party_name in parties:
164         dict_words["Party of speaker:"] = party_name
165     else:
166         dict_words["Party of speaker:"] = "?"
167
168     # Nachname mit Liste weiblicher Abgeordneter abgleichen und
    Geschlecht ins Dictionary eintragen
```

```
169     surname = word_tokens_name_party[-2]
170     if len(word_tokens_name_party) == 1:
171         return
172     if surname in female_MPs:
173         dict_words["Gender of speaker:"] = "w"
174     else:
175         dict_words["Gender of speaker:"] = "m"
176 
177     # männliche, weibliche, neutrale Wörter mit Listen
    abgleichen und in Dictionary eintragen
178     word_tokens_speech = single_speakers_texts[1].split()
179     for word in word_tokens_speech:
180         dict_words["Total words:"] += 1
181         if word in male_words:
182             dict_words["Male words:"] += 1
183         elif word in female_words:
184             dict_words["Female words:"] += 1
185         elif word in neutral_words:
186             dict_words["Neutral words:"] += 1
187     return dict_words
188 
189 
190 # sucht in SoupObject nach Nachname, Fraktion/Partei und dem
    Sprechertext
191 def extract_info_from_soup(single_speeches):
192     # findet alle XML-Tags <nachname>
193     speaker = single_speeches.find("nachname")
194     # findet alle XML-Tags <fraktion> und erstellt einen String,
    der den Partei-/Fraktionsnamen enthält
195     party = single_speeches.find_all("fraktion")
196     for element in single_speeches.find_all("fraktion"):
197         party = str(element.text)
198     # filtert den Redetext anhand der für "Klasse" vergebenen
    Attribute über Dictionary (siehe Strukturdefinition der
199     # Protokolle)
200     speech = single_speeches.find_all("p", {"klasse": ("J_1", "J
    ", "Z", "O")})
201     return speaker, party, speech
202 
203 
204 # Erstellung des Dictionarys, in dem Geschlecht und Partei der
    Sprechenden als auch deren Verwendung der
```

```
205 # Substantive und Gesamtlänge des Sprechertextes notiert wird
206 def create_dictionary_from_soup(parties, female_MPs, male_words,
        female_words, neutral_words, analyse):
207     # erstellt Dictionary mit gleichem Aufbau wie oben, damit
    Zusammenführung möglich ist
208     dict_words = {"Gender of speaker:": " ", "Party of speaker:"
    : " ", "Male words:": 0, "Female words:": 0,
209                   "Neutral words:": 0, "Total words:": 0}
210     word_tokens_party = str(analyse[1])
211     get_party = word_tokens_party.split()
212     party_name = get_party[0]
213     if party_name in parties:
214         dict_words["Party of speaker:"] = party_name
215     else:
216         dict_words["Party of speaker:"] = "?"
217
218     # Nachname mit Liste weiblicher Abgeordneter abgleichen und
    Geschlecht ins Dictionary eintragen
219     soup_object_surname = str(analyse[0])
220     word_tokens_surname = soup_object_surname.split()
221     if len(word_tokens_surname) < 2:
222         return
223     surname = word_tokens_surname[1]
224     if surname in female_MPs:
225         dict_words["Gender of speaker:"] = "w"
226     else:
227         dict_words["Gender of speaker:"] = "m"
228
229     # männliche, weibliche, neutrale Wörter mit Listen
    abgleichen und in Dictionary eintragen
230     soup_object_speech = str(analyse[2])
231     word_tokens_speech = soup_object_speech.split()
232     for word in word_tokens_speech:
233         # zählt Gesamtwortanzahl pro Text, damit am Ende
    relative Werte verglichen werden können
234         dict_words["Total words:"] += 1
235         if word in male_words:
236             dict_words["Male words:"] += 1
237         elif word in female_words:
238             dict_words["Female words:"] += 1
239         elif word in neutral_words:
240             dict_words["Neutral words:"] += 1
```

```
241     return dict_words
242
243
244 # Abspeichern des Dictionarys als DataFrame zur
        Weiterverarbeitung
245 def save_dictionary_as_df(dict_words):
246     data_frame = pd.DataFrame(dict_words)
247     # tauscht Zeilen und Spalten
248     data_frame = data_frame.T
249     # füllt Lücken mit 0
250     data_frame.fillna(0, inplace=True)
251     # speichert den gesamten DataFrame, damit Rohdaten verfügbar
        bleiben
252     with open("HausarbeitRohdaten.csv", "w", encoding="utf-8")
        as outfile:
253         data_frame.to_csv(outfile, sep=",")
254     return data_frame
255
256
257 # liest DataFrame mit Rohdaten zur Weiterverarbeitung ein
258 def read_table():
259     with open("HausarbeitRohdaten.csv", "r", encoding="utf-8")
        as infile:
260         table_data = pd.read_csv(infile, sep=",", index_col=0)
261     return table_data
262
263
264 # führt Berechnungen durch, die für Erstellung des
        Liniendiagramms benötigt werden (Summen, Prozentwerte)
265 def calculations_for_lineplot(table_data):
266     # summiert die absolute Anzahl von male/female/neutral words
        pro Periode zur Weiterverarbeitung in neuen Zeilen auf
267     table_data.loc["Gesamt Periode 01, absolut"] = np.sum(
        table_data.loc["Anfang Periode 01": "Anfang Periode 07",
268                                                         "Male
        words:": "Total words:"], axis=0)
269     table_data.loc["Gesamt Periode 07, absolut"] = np.sum(
        table_data.loc["Anfang Periode 07": "Anfang Periode 14",
270                                                         "Male
        words:": "Total words:"], axis=0)
271     table_data.loc["Gesamt Periode 14, absolut"] = np.sum(
        table_data.loc["Anfang Periode 14": "Anfang Periode 19",
```

```
272                                                                 "Male
    words:": "Total words:"], axis=0)
273     table_data.loc["Gesamt Periode 19, absolut"] = np.sum(
    table_data.loc["Anfang Periode 19": "ENDE",
274                                                                 "Male
    words:": "Total words:"], axis=0)
275     # berechnet relative Häufigkeit der Wörter pro Periode
276     table_data.loc["Gesamte Periode 01, relativ (in %)"] =
    table_data.loc["Gesamt Periode 01, absolut",
277                                                                 "Male
     words:": "Neutral words:"].div(table_data.loc
278
                                        [
279
                                            "Gesamt Periode 01,
    absolut", "Total words:"])
280     table_data.loc["Gesamte Periode 07, relativ (in %)"] =
    table_data.loc["Gesamt Periode 07, absolut",
281                                                                 "Male
     words:": "Neutral words:"].div(table_data.loc
282
                                        [
283
                                            "Gesamt Periode 07,
    absolut", "Total words:"])
284     table_data.loc["Gesamte Periode 14, relativ (in %)"] =
    table_data.loc["Gesamt Periode 14, absolut",
285                                                                 "Male
     words:": "Neutral words:"].div(table_data.loc
286
                                        [
287
                                            "Gesamt Periode 14,
    absolut", "Total words:"])
288     table_data.loc["Gesamte Periode 19, relativ (in %)"] =
    table_data.loc["Gesamt Periode 19, absolut",
289                                                                 "Male
     words:": "Neutral words:"].div(table_data.loc
290
                                        [
```

```
291
                                        "Gesamt Periode 19,
    absolut", "Total words:"])
292     # reduziert Tabelle auf die neu erstellten Zeilen
293     table_data_lineplot = table_data.loc["Gesamt Periode 01,
    absolut":"Gesamte Periode 19, relativ (in %)", :]
294     with open("Daten für Lineplot.csv", "w", encoding="utf8") as
     outfile:
295         table_data_lineplot.to_csv(outfile, sep=",")
296     return table_data_lineplot
297
298
299 # Erstellung des Liniendiagramms
300 def make_lineplot_words_used(table_data_lineplot):
301     # legt Grundsätzliches fest: Stil, Titel,
    Achsenbeschriftungen
302     line_chart = pygal.Line(style=CleanStyle, legend_at_bottom=
    True,
303                             legend_at_bottom_columns=3)
304     # # entkommentieren für Diagrammtitel
305     # line_chart.title = "Verwendung verschiedener Substantive
    je Wahlperiode"
306     line_chart.y_title = "Relative Häufigkeit (in %)"
307     line_chart.x_labels = ("Periode 1", "Periode 7", "Periode 14
    ", "Periode 19")
308     # Daten aus der Tabelle in line plot einpflegen
309     line_chart.add("Männliche Substantive", table_data_lineplot.
    loc["Gesamte Periode 01, relativ (in %)":
310
        "Gesamte Periode 19, relativ (in %)",
311                                         "Male words:"])
312     line_chart.add("Weibliche Substantive", table_data_lineplot.
    loc["Gesamte Periode 01, relativ (in %)":
313
        "Gesamte Periode 19, relativ (in %)",
314                                         "Female words:"])
315     line_chart.add("Neutrale Substantive", table_data_lineplot.
    loc["Gesamte Periode 01, relativ (in %)":
316
       "Gesamte Periode 19, relativ (in %)",
317                                        "Neutral words:"])
318     # Speichern der Datei als .svg
```

```
319     line_chart.render_to_file("Frequency-MFN-words-Periods.svg")
320
321
322 # führt Berechnungen durch, die für Erstellung des ersten
    Balkendiagramms benötigt werden (Summen, Prozentwerte)
323 def calculations_for_barchart_speakers(table_data):
324     # sammelt Daten der männlichen Sprecher, summiert sie auf
    und ermittelt relative Häufigkeit
325     table_data.loc["Male speakers, absolute use"] = table_data.
    loc[table_data["Gender of speaker:"] == "m",
326
          ["Male words:", "Female words:", "Neutral words:",
327
           "Total words:"]].sum()
328     table_data.loc["Male speakers, relative use (in %)"] =
    table_data.loc["Male speakers, absolute use", "Male words:":
329
                                                "Neutral words:"].
    div(
330         table_data.loc["Male speakers, absolute use", "Total
    words:"])
331
332     # sammelt Daten der weiblichen Sprecher, summiert sie auf
    und ermittelt relative Häufigkeit
333     table_data.loc["Female speakers, absolute use"] = table_data
    .loc[table_data["Gender of speaker:"] == "w",
334
          ["Male words:", "Female words:", "Neutral words:",
335
           "Total words:"]].sum()
336     table_data.loc["Female speakers, relative use (in %)"] =
    table_data.loc["Female speakers, absolute use",
337                                                              "
    Male words:": "Neutral words:"].div(
338         table_data.loc["Female speakers, absolute use", "Total
    words:"])
339
340     # reduziert die Tabelle auf essenzielle Zeilen und Spalten
341     table_data_barchart_speakers = table_data.loc["Male speakers
    , absolute use": "Female speakers, relative use (" \
342
                      "in %)", :]
```

```
343
344     with open("Daten für Barchart Sprecher.csv", "w", encoding="
    utf8") as outfile:
345         table_data_barchart_speakers.to_csv(outfile, sep=",")
346     return table_data_barchart_speakers
347
348
349 # erstellt Balkendiagramm
350 def make_barchart_speakers(table_data_barchart_speakers):
351     # legt Grundsätzliches fest: Stil, Titel,
    Achsenbeschriftungen
352     bar_chart = pygal.Bar(style=CleanStyle, legend_at_bottom=
    True)
353     # # entkommentieren für Diagrammtitel
354     # bar_chart.title = "Verwendung verschiedener Substantive in
     Abhängigkeit zu Geschlecht"
355     bar_chart.y_title = "Relative Häufigkeit (in %)"
356     bar_chart.x_labels = ("Männliche Substantive", "Weibliche
    Substantive", "Neutrale Substantive")
357
358     # Daten aus Tabelle in line chart einpflegen
359     bar_chart.add("Männliche Sprecher", [
    table_data_barchart_speakers.loc["Male speakers, relative use
     (in %)",
360
              "Male words:"],
361
    table_data_barchart_speakers.loc[
362                                          "Male speakers,
    relative use (in %)", "Female words:"],
363
    table_data_barchart_speakers.loc[
364                                          "Male speakers,
    relative use (in %)", "Neutral words:"]])
365
366     bar_chart.add("Weibliche Sprecher", [
    table_data_barchart_speakers.loc["Female speakers, relative
    use (in %)",
367
              "Male words:"],
368
    table_data_barchart_speakers.loc[
```

```
369                                                 "Female speakers,
    relative use (in %)", "Female words:"],
370
    table_data_barchart_speakers.loc[
371                                                 "Female speakers,
    relative use (in %)", "Neutral words:"]])
372     # Speichern der Datei als .svg
373     bar_chart.render_to_file("Distribution-MFN-words-Gender.svg"
    )
374
375
376# führt Berechnungen durch, die für Erstellung des zweiten
    Balkendiagramms benötigt werden (Summen, Prozentwerte)
377def calculations_for_bar_chart_parties(table_data):
378     # je Parteien/Fraktionen im Bundestag: sammelt Daten aus
    Tabelle, summiert sie und berechnet relative
379     # Häufigkeit (analog zu Verfahren bei männlichen und
    weiblichen Redner/-innen)
380
381     # SPD
382     table_data.loc["SPD"] = table_data.loc[table_data["Party of
    speaker:"] == "SPD",
383                                            ["Male words:", "
    Female words:", "Neutral words:",
384                                             "Total words:"]].sum
    ()
385     table_data.loc["(SPD):"] = table_data.loc[table_data["Party
    of speaker:"] == "(SPD):",
386                                               ["Male words:", "
    Female words:", "Neutral words:",
387                                                "Total words:"]].
    sum()
388     table_data.loc["SPD speakers"] = np.sum(table_data.loc["SPD"
    : "(SPD):", :], axis=0)
389     table_data.loc["SPD speakers, relative use (in %)"] = (
390         table_data.loc["SPD speakers", "Male words:": "Neutral
    words:"].div(
391             table_data.loc["SPD speakers", "Total words:"]))
392
393     # CDU
394     table_data.loc["(CDU):"] = table_data.loc[table_data["Party
    of speaker:"] == "(CDU):",
```

```
395                                                 ["Male words:", "
    Female words:", "Neutral words:",
396                                                  "Total words:"]].
    sum()
397     table_data.loc["(CSU):"] = table_data.loc[table_data["Party
    of speaker:"] == "(CSU):",
398                                                 ["Male words:", "
    Female words:", "Neutral words:",
399                                                  "Total words:"]].
    sum()
400     table_data.loc["(CDU und CSU):"] = table_data.loc[table_data
    ["Party of speaker:"] == "(CDU und CSU):",
401                                                       ["Male
    words:", "Female words:", "Neutral words:",
402                                                        "Total
    words:"]].sum()
403     table_data.loc["(CDU/CSU):"] = table_data.loc[table_data["
    Party of speaker:"] == "(CDU/CSU):",
404                                                     ["Male words:"
    , "Female words:", "Neutral words:",
405                                                      "Total words:
    "]].sum()
406     table_data.loc["(BP):"] = table_data.loc[table_data["Party
    of speaker:"] == "(BP):",
407                                                 ["Male words:", "
    Female words:", "Neutral words:",
408                                                  "Total words:"]].
    sum()
409     table_data.loc["CDU"] = table_data.loc[table_data["Party of
    speaker:"] == "CDU",
410                                               ["Male words:", "
    Female words:", "Neutral words:", "Total words:"]].sum()
411     table_data.loc["CSU"] = table_data.loc[table_data["Party of
    speaker:"] == "CSU",
412                                               ["Male words:", "
    Female words:", "Neutral words:", "Total words:"]].sum()
413     table_data.loc["CDU und CSU"] = table_data.loc[table_data["
    Party of speaker:"] == "CDU und CSU",
414                                                     ["Male words:
    ", "Female words:", "Neutral words:",
415                                                      "Total words
    :"]].sum()
```

```
416     table_data.loc["CDU/CSU"] = table_data.loc[table_data["Party
        of speaker:"] == "CDU/CSU",
417                                                ["Male words:", "
    Female words:", "Neutral words:", "Total words:"]].sum()
418     table_data.loc["BP"] = table_data.loc[table_data["Party of
    speaker:"] == "BP",
419                                               ["Male words:", "
    Female words:", "Neutral words:", "Total words:"]].sum()
420     table_data.loc["CDU speakers"] = np.sum(table_data.loc["(CDU
    ):": "BP", :], axis=0)
421     table_data.loc["CDU speakers, relative use (in %)"] = (
422         table_data.loc["CDU speakers", "Male words:": "Neutral
    words:"].div(
423             table_data.loc["CDU speakers", "Total words:"]))
424
425     # BÜNDNIS 90/DIE GRÜNEN
426     table_data.loc["GRÜNEN):"] = table_data.loc[table_data["
    Party of speaker:"] == "GRÜNEN):",
427                                                 ["Male words:",
    "Female words:", "Neutral words:",
428                                                  "Total words:"
    ]].sum()
429     table_data.loc["BÜNDNIS 90/DIE GRÜNEN"] = table_data.loc[
    table_data["Party of speaker:"] == "BÜNDNIS",
430                                                              ["
    Male words:", "Female words:", "Neutral words:",
431                                                               "
    Total words:"]].sum()
432     table_data.loc["Grünen speakers"] = np.sum(table_data.loc["
    GRÜNEN):": "BÜNDNIS 90/DIE GRÜNEN", :],
433                                                axis=0)
434     table_data.loc["DIE GRÜNEN speakers, relative use (in %)"] =
    (table_data.loc["Grünen speakers", "Male words:":
435
436                                    "Neutral words:"].div(
        table_data.loc["Grünen speakers", "Total words:"]))
437
438     # Die Linke
439     table_data.loc["LINKE):"] = table_data.loc[table_data["Party
        of speaker:"] == "LINKE):",
440                                                ["Male words:", "
    Female words:", "Neutral words:",
```

```
441                                                    "Total words:"
    ]].sum()
442     table_data.loc["(PDS):"] = table_data.loc[table_data["Party
    of speaker:"] == "(PDS):",
443                                                   ["Male words:", "
    Female words:", "Neutral words:",
444                                                    "Total words:"]].
    sum()
445     table_data.loc["DIE LINKE"] = table_data.loc[table_data["
    Party of speaker:"] == "DIE",
446                                                     ["Male words:",
     "Female words:", "Neutral words:",
447                                                      "Total words:"
    ]].sum()
448     table_data.loc["Linke speakers"] = np.sum(table_data.loc["
    LINKE):": "DIE LINKE", :],
449                                                  axis=0)
450     table_data.loc["DIE LINKE speakers, relative use (in %)"] =
    (table_data.loc["Linke speakers", "Male words:":
451
                                          "Neutral words:"].div(
452         table_data.loc["Linke speakers", "Total words:"]))
453
454     # AfD
455     table_data.loc["(AfD):"] = table_data.loc[table_data["Party
    of speaker:"] == "(AfD):",
456                                                   ["Male words:", "
    Female words:", "Neutral words:",
457                                                    "Total words:"]].
    sum()
458     table_data.loc["AfD"] = table_data.loc[table_data["Party of
    speaker:"] == "AfD",
459                                                 ["Male words:", "
    Female words:", "Neutral words:",
460                                                  "Total words:"]].sum
    ()
461     table_data.loc["AfD speakers"] = np.sum(table_data.loc["(AfD
    ):": "AfD", :], axis=0)
462     table_data.loc["AfD speakers, relative use (in %)"] = (
    table_data.loc["AfD speakers", "Male words:":
463
                                "Neutral words:"].div(
```

```
464         table_data.loc["AfD speakers", "Total words:"]))
465
466     # FDP
467     table_data.loc["(FDP):"] = table_data.loc[table_data["Party
    of speaker:"] == "(FDP):",
468                                               ["Male words:", "
    Female words:", "Neutral words:",
469                                                "Total words:"]].
    sum()
470     table_data.loc["(F D P ):"] = table_data.loc[table_data["
    Party of speaker:"] == "(F D P ):",
471                                                  ["Male words:",
     "Female words:", "Neutral words:",
472                                                   "Total words:"
    ]].sum()
473     table_data.loc["FDP"] = table_data.loc[table_data["Party of
    speaker:"] == "FDP",
474                                            ["Male words:", "
    Female words:", "Neutral words:",
475                                             "Total words:"]].sum
    ()
476     table_data.loc["F D P "] = table_data.loc[table_data["Party
    of speaker:"] == "F D P ",
477                                               ["Male words:", "
    Female words:", "Neutral words:",
478                                                "Total words:"]].
    sum()
479     table_data.loc["FDP speakers"] = np.sum(table_data.loc["(FDP
    ):": "F D P ", :],
480                                             axis=0)
481     table_data.loc["FDP speakers, relative use (in %)"] = (
    table_data.loc["FDP speakers", "Male words:":
482
                               "Neutral words:"].div(
483         table_data.loc["FDP speakers", "Total words:"]))
484
485     # reduziert Tabelle auf essenzielle Zeilen und Spalten
486     table_data_barchart_parties = table_data.loc["SPD": "FDP
    speakers, relative use (in %)",
487                                  "Male words:": "Total words:"]
488     with open("Daten für Barchart Partei.csv", "w", encoding="
    utf8") as outfile:
```

```
489         table_data_barchart_parties.to_csv(outfile, sep=",")
490     return table_data_barchart_parties
491
492
493 # erstellt Balkendiagramm
494 def make_barchart_parties(table_data_barchart_parties):
495     # legt Grundsätzliches fest: Stil, Titel und
    Achsenbeschriftung
496     bar_chart = pygal.Bar(style=custom_style, legend_at_bottom=
    True)
497     # # entkommentieren für Diagrammtitel
498     # bar_chart.title = "Verwendung verschiedener Substantive in
     Abhängigkeit zu Parteizugehörigkeit"
499     bar_chart.y_title = "Relative Häufigkeit (in %)"
500     bar_chart.x_labels = ("Männliche Substantive", "Weibliche
    Substantive", "Neutrale Substantive")
501
502     # Daten aus Tabelle für jede Partei/Fraktion in line chart
    einpflegen
503     bar_chart.add("DIE LINKE", [table_data_barchart_parties.loc[
    "DIE LINKE speakers, relative use (in %)",
504
    "Male words:"],
505                                 table_data_barchart_parties.loc[
    "DIE LINKE speakers, relative use (in %)",
506
    "Female words:"],
507                                 table_data_barchart_parties.loc[
    "DIE LINKE speakers, relative use (in %)",
508
    "Neutral words:"]])
509
510     bar_chart.add("SPD", [table_data_barchart_parties.loc["SPD
    speakers, relative use (in %)", "Male words:"],
511                           table_data_barchart_parties.loc["SPD
    speakers, relative use (in %)", "Female words:"],
512                           table_data_barchart_parties.loc["SPD
    speakers, relative use (in %)", "Neutral words:"]])
513
514     bar_chart.add("DIE GRÜNEN", [table_data_barchart_parties.loc
    ["DIE GRÜNEN speakers, relative use (in %)",
```

```
515
    "Male words:"],
516                                 table_data_barchart_parties.loc
    [
517                                     "DIE GRÜNEN speakers,
    relative use (in %)", "Female words:"],
518                                 table_data_barchart_parties.loc
    [
519                                     "DIE GRÜNEN speakers,
    relative use (in %)", "Neutral words:"]])
520
521     bar_chart.add("CDU/CSU", [table_data_barchart_parties.loc["
    CDU speakers, relative use (in %)", "Male words:"],
522                               table_data_barchart_parties.loc["
    CDU speakers, relative use (in %)", "Female words:"],
523                               table_data_barchart_parties.loc["
    SPD speakers, relative use (in %)", "Neutral words:"]])
524
525     bar_chart.add("FDP", [table_data_barchart_parties.loc["FDP
    speakers, relative use (in %)", "Male words:"],
526                           table_data_barchart_parties.loc["FDP
    speakers, relative use (in %)", "Female words:"],
527                           table_data_barchart_parties.loc["FDP
    speakers, relative use (in %)", "Neutral words:"]])
528
529     bar_chart.add("AfD", [table_data_barchart_parties.loc["AfD
    speakers, relative use (in %)", "Male words:"],
530                           table_data_barchart_parties.loc["AfD
    speakers, relative use (in %)", "Female words:"],
531                           table_data_barchart_parties.loc["AfD
    speakers, relative use (in %)", "Neutral words:"]])
532
533     # speichert Diagramm als .svg-Datei
534     bar_chart.render_to_file("Distribution-MFN-words-Party.svg")
535
536
537 def main():
538     # leeres Dictionary anlegen
539     speeches_data = {}
540
541     # Listen einlesen
542     female_MPs = read_list("Female_MPs.txt")
```

```
543     parties = read_list("Parties.txt")
544     male_words = read_list("Male_Words.txt")
545     female_words = read_list("Female_Words.txt")
546     neutral_words = read_list("Neutral_Words.txt")
547     parties_for_soup = read_list("Parties_for_Soup.txt")
548 
549     for textfile in filename_list:
550         # markiert Periode des Textes
551         text_period_beginning = "Anfang Periode " + basename(
    textfile).split(".xml")[0][0:2]
552 
553         # baut Einträge in Dictionary ein, die später als
    Orientierung und zum Slicing des DataFrames dienen
554         speeches_data[text_period_beginning] = {"Gender of
    speaker:": 0, "Party of speaker:": 0, "Male words:":
555             0, "Female words:": 0, "Neutral words:": 0, "Total
    words:": 0}
556         text = read_textfile(textfile)
557         cleaned_text = clean_texts(text)
558 
559         # ruft Liste mit Positionsangaben der Parteinamen auf
560         split_at_place = split_text_new_speaker(cleaned_text)
561 
562         # Laufvariable initialisieren
563         i = 0
564 
565         while i < len(split_at_place):
566             # single_speaker_text ist Redebeitrag einzelner
    Person beginnend mit deren Namen (ab Zeilenumbruch) bis
567             # zum nächsten Redebeitrag
568             single_speaker_text = find_start_of_line(
    cleaned_text, split_at_place, i)
569 
570             # führt Index ein, bei dem jeder Textabschnitt
    nummeriert wird für spätere Überführung in Data Frame
571             text_file_name = basename(textfile).split(".xml")[0]
     + "-" + str(i + 1)
572             if single_speaker_text is not None:
573                 word_dict = create_dictionary(parties,
    female_MPs, male_words, female_words, neutral_words,
574 
    single_speaker_text)
```

```
575                 # für jeden einzelnen Redetext wird eine
    Tabellenspalte erstellt und dem Redetext über Index
    zugeordnet
576                 speeches_data[text_file_name] = word_dict
577             i += 1
578
579     for file in filename_list_period_19:
580         text = read_textfile(file)
581         text_cleaned = clean_texts(text)
582
583         # SoupObject erstellen aus bereinigtem Text mit XML-
    Parser
584         soup = BeautifulSoup(text_cleaned, 'lxml')
585
586         # SoupObject durchsuchen nach Tag <rede>
587         soup_single_speeches = soup.find_all("rede")
588         i = 0
589
590         text_period_beginning = "Anfang Periode " + basename(
    file).split(".xml")[0][0:2]
591         # baut Eintrag in Dictionary ein, der später als
    Orientierung und zum Slicing des DataFrames dient
592         speeches_data[text_period_beginning] = {"Gender of
    speaker:": 0, "Party of speaker:": 0, "Male words:": 0,
593                                                "Female words:":
     0, "Neutral words:": 0, "Total words:": 0}
594         while i < len(soup_single_speeches):
595             analyse = extract_info_from_soup(
    soup_single_speeches[i])
596             word_dict = create_dictionary_from_soup(
    parties_for_soup, female_MPs, male_words, female_words,
597
    neutral_words, analyse)
598             # führt Index ein, bei dem jeder Textabschnitt
    nummeriert wird für spätere Überführung in Data Frame
599             text_file_name = basename(file).split(".xml")[0] + "
    -" + str(i + 1)
600             speeches_data[text_file_name] = word_dict
601             i += 1
602     speeches_data["ENDE"] = {"Gender of speaker:": 0, "Party of
    speaker:": 0, "Male words:": 0, "Female words:": 0,
```

```
603                                 "Neutral words:": 0, "Total words:"
    : 0}
604
605     save_dictionary_as_df(speeches_data)
606     csv_doc = read_table()
607     total_numbers_per_period = calculations_for_lineplot(csv_doc
    )
608     make_lineplot_words_used(total_numbers_per_period)
609     gender_distribution = calculations_for_barchart_speakers(
    csv_doc)
610     make_barchart_speakers(gender_distribution)
611     party_distribution = calculations_for_bar_chart_parties(
    csv_doc)
612     make_barchart_parties(party_distribution)
613
614
615 filename_list = glob.glob("Periode_*/*.xml")
616 filename_list_period_19 = glob.glob("Periode19/*.xml")
617
618 if __name__ == "__main__":
619     main()
```

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich diese Hausarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht habe. Die Arbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher auch nicht veröffentlicht.

Trier, den 14. September 2021