Friedrich-Alexander-Universität

Philosophische Fakultät und Fachbereich Theologie

Lehrstuhl für Korpus- und Computerlinguistik

Bachelorarbeit im Fach Computerlinguistik

Analyse der Social-Media-Kommunikationsstrategien von Trump und Musk mittels CADS und FlexiConc

Vivien Müller

Sommersemester 2025

Bearbeitungszeit: 21.07.2025 - 28.10.2025

Betreuerin: Prof. Dr. Stephanie Evert

Studiengang: Philosophie & Computerlinguistik

Fachsemester: 8

Matrikelnummer: 22887293

Email: vivien.v.mueller@fau.de

Inhaltsverzeichnis

1. Einleitung

“China is eating our lunch”. Diese und viele weitere interessante Redewendungen postet Donald Trump täglich auf der Social Media Plattform „Truth Social“(Polak, 2024). Neben abgewandelten Redewendungen und der Verwendung sprachlicher Bilder ist wahrscheinlich besonders die Verwendung repetitiver Muster charakteristisch für Trump. Welcher weiteren Strategien sich Donald Trump auf Truth Social und Twitter[1] bedient, stellt die Fragestellung dieser Bachelorarbeit dar. Neben Trump wird außerdem der zeitweise reichste Mensch der Welt (*Elon Musks Vermögen Erreicht 500 Milliarden Dollar*, 2025), Elon Musk analysiert. Seine Unternehmen stehen unter den humanitären Zielen der Überlebenssicherung durch Expansion ins Weltall mit SpaceX, Unterstützung erneuerbarer Energien durch Tesla, Ausgleich von Gehirnverletzungen durch Neuralink und Redefreiheit durch den Erwerb Twitters (Miklaszewicz, 2023). Aufgrund der Bekanntheit und Entscheidungsgewalt beider Männer haben die Social-Media-Posts der beiden einen großen Einfluss auf die amerikanische Gesellschaft und darüber hinaus auch auf die Welt. Zum Beispiel ist Musk neben Trump stark in den Ukrainekrieg involviert, da er diverse Satelliten des Unternehmens SpaceX für das ukrainische Militär zur Kommunikation zur Verfügung stellt (Farrow, 2023). Auch Trump hat als Präsident Amerikas einen großen Einfluss auf innen- wie auch außenpolitische Entscheidungen (Polak, 2024).

Aufgrund der weltpolitischen Relevanz beider Männer sollen in meiner Bachelorarbeit mit dem Titel „Analyse der Social-Media-Kommunikationsstrategien von Trump und Musk mittels CADS (corpus-assisted discourse studies) und FlexiConc“ verschiedene Kommunikationsstrategien Trumps und Musks auf Truth Social und Twitter untersucht werden.

Für die Analyse bedarf es zunächst qualitativ hochwertiger Daten, welche die Grundlage einer guten Analyse darstellen. Elon Musks Social Media Posts von 2010 bis April 2025 sind bereits auf Kaggle zum Download als csv-file verfügbar. Diese Daten werden getaggt und entsprechend vorverarbeitet, um sie anschließend bei der IMS Corpusworkbench (CWB) als Korpus einzulesen. Auch ein Großteil der Tweets von Donald Trump (2009–2024) liegt im Vorfeld in aufbereiteter Form vor. Um die aktuellsten Beiträge ab November 2024 zu erfassen, erfolgte zusätzlich Scraping der Website *Rollcall* mithilfe des Tools Playwright, wodurch auch Tweets vom 4. November 2024 bis zum 25. August 2025 in das Datenset ergänzt werden können. Für die Vergabe von POS-Tags wurden zunächst mehrere Tagger anhand eines Testdatensatzes von 50 Tweets verglichen. Unter diesen wurde dann der beste Tagger ausgewählt. Im Anschluss daran erfolgt die Analyse beider Korpora TRUMP und MUSK mittels FlexiConc, einem Korpus-Analyse-Tool und cwb-ccc im Hinblick auf verschiedene Kommunikationsstrategien. Die verwendete Methodik ist eine Kombination von „close reading“ und „distant reading“ im Rahmen von CADS (corpus-assisted discourse studies). Dabei steht hauptsächlich die Analyse von Konkordanzen mittels

[1] Seit dem 27.10.2022 ist Musk CEO der Plattform (Miklaszewicz, 2023). Twitter ist seit dem 24. Juli 2023 als „X“ bekannt. Ich werde im weiteren Verlauf der Übersichtlichkeit halber von Twitter sprechen.

FlexiConc im Fokus, ergänzt durch Frequenz- und Keywordanalysen über die Bildung von Subkorpora mit cwb-ccc. Abschließend folgt ein Fazit zur Verwendung von FlexiConc und der Analyse der Kommunikationsstrategien. Aufgrund der umfangreichen und zeitaufwändigen Datensammlung und -aufbereitung nimmt die Analyse der Kommunikationsstrategien nun einen geringeren Teil der Bachelorarbeit ein.

2. Der aktuelle Stand der Forschung

In folgendem Abschnitt wird es der aktuelle Stand der Forschung in Bezug auf die wichtigsten Bereiche der Bachelorarbeit - nämlich Web Scraping, POS-Tagging, CADS und bekannte Korpus-Analyse-Tools - behandelt.

2.1 Web Scraping

Oft wird für linguistische Untersuchungen kein vollständig vorverarbeitetes Korpus zur Verfügung gestellt, sodass der Forschende vor der Herausforderung steht, sich geeignete Daten zu beschaffen. Dies ist zum Beispiel mit Web Scraping möglich, wenn geeignete Daten im Internet vorhanden sind (Khder, 2021). Web Scraping oder Web Crawling beschreibt das gezielte, automatische Extrahieren bestimmter Daten von Websites mithilfe ausgewählter Software (El Asikri et al., 2020; Gudavalli & JayaLakshmi, 2025; Khder, 2021). Dabei werden die beiden Begriffe Web Scraping oder Web Crawling meist synonym verwendet, wobei Web Crawling eher das Erfassen der Daten aus dem Internet meint und Web Scraping den gesamten Prozess der Datenextraktion sowie das Herunterladen und Speichern der Daten umfasst (Khder, 2021). Geläufig sind außerdem die Begriffe Data Mining oder Data Harvesting (Gudavalli & JayaLakshmi, 2025). Web Scraping wird meistens eher in Marketingumgebungen verwendet, um Produktpreise, Preisschwankungen oder Marktentwicklungen zu beobachten und auszuwerten (Gudavalli & JayaLakshmi, 2025). Auch im Bereich der Korpuslinguistik kann auf Web Scraping zurückgegriffen werden, um Daten aus dem Internet als linguistische Quelle zu erhalten, was meistens mit Duplikat-entfernung, Reinigen und Annotieren der Daten einher geht (Barbaresi, 2021). Insgesamt gibt es eine Vielzahl verschiedener Scraper in den verschiedensten Programmiersprachen, sodass zunächst gut ausgewählt werden muss, welches Modell für die gewünschten Aufgaben in Frage kommt. Lxml etwa wurde für die Programmiersprache C geschrieben und Puppeteer beispielsweise für JavaScript (Abodayeh et al., 2023). Da sich Python am besten zur Sprachverarbeitung eignet, weil es die meisten sprachbezogenen Bibliotheken zur Verfügung stellt (Abodayeh et al., 2023; Bale et al., 2022), werden im Folgenden nur die bekanntesten Scraper und Parser für Python vorgestellt. Zu diesen zählen Beautifulsoup, Scrapy und Selenium. Playwright hingegen ist einer der unbekannteren Scraper, sodass bisher nur wenige Paper zu Playwright und vor allem zu dessen Anwendung im Bereich der Computerlinguistik verfasst wurden. Hierbei lassen sich vor allem zwei Arten an Scrapern unterscheiden. Diejenigen, die für statische Webseiten geeignet sind und diejenigen, die auch dynamische Webseiten verarbeiten können. Dynamische Webseiten werden meist mit Java-Script nachgeladen und sind nicht auf einen Blick vollständig verfügbar, sondern laden erst – beispielsweise Scrollen – Stück für Stück weiter (Gudavalli & JayaLakshmi, 2025). Abgesehen

davon unterscheiden sich die Scraper in einer Vielzahl verschiedener Merkmale, sodass bei jeder Anwendung individuell geprüft werden sollte, welcher Scraper sich am besten eignet. Sowohl Beautifulsoup als auch Selectolax und Scrapy sind nur für statische Webseiten (Gudavalli & JayaLakshmi, 2025), die die Inhalte der Webseite als vollständiges HTML-Dokument zur Verfügung stellen, geeignet. Beautifulsoup ist eine sehr bekannte Pythonbibliothek, die effizient und einsteigerfreundlich konzipiert wurde (Persson, 2019). Allerdings ist Beautifulsoup ein Parser, der ausschließlich für statische Webseiten wie zum Beispiel Adidas oder Amazon geeignet ist (Abodayeh et al., 2023). Mehr zu den Anwendungsbereichen und Vorteilen von Beautifulsoup wird im Paper „Web Scraping for Data Analytics – A Beautiful Soup Implementation" von Abodayeh et al. (2023) beschrieben. Selectolax ist ein schnelles Scraping Tool mit CSS-Selektoren, das nicht so anwenderfreundlich wie Beautifulsoup ist, dafür allerdings flexibler mit dem Browser umgehen kann (Rooney, 2022). Heutzutage lässt sich der zunehmende Einsatz von KI und Bots im Netz beobachten. 46% der Anfragen im Internet werden allein durch Bots gestellt – mit steigender Tendenz (Gul et al., 2025). Aufgrund stärkerer Datenschutzbestimmungen und Botblockern entwickelt sich auch die Art und Weise, wie Webseiten konzipiert werden, fließend weiter. Es gibt immer häufiger Captcha-Abfragen und andere Fallen (z.B. sog. honeypots), die Bots davon abhalten sollen, entsprechende Internetseiten zu durchsuchen und gegebenenfalls Informationen zu extrahieren (Gul et al., 2025). Dies ist nicht nur aufgrund des Datenschutzes nötig, sondern auch, weil zu viele Anfragen durch Bots – wie etwa bei Wikipedia – die Server überlasten (Gul et al., 2025). Dadurch wird auch die stetige Anpassung der Scraper erforderlich, um mit neueren Gegebenheiten umgehen zu können (Gul et al., 2025). Um Bot Blocker und andere Maßnahmen effizient zu umgehen, ist es notwendig, dass der Scraper sich menschenähnlich verhalten kann, indem er beispielsweise nicht zu viele Anfragen kurz nacheinander an Internetseiten stellt. Zudem müssen die Scraper vermehrt für Webseiten ausgelegt sein, die dynamisch über JavaScript nachgeladen werden, wenn der Nutzer durch die Seiten iteriert oder scrollt (Gudavalli & JayaLakshmi, 2025). Selenium und Playwright sind beispielsweise solche Tagger, die für dynamische Seiten geeignet sind. Selenium ist dabei der bekanntere Scraper von beiden. Ursprünglich wurde es nicht für das Scraping von Webseiten konzipiert, sondern um als web basiertes Online-Tool Testen und Fehleranalyse von Webseiten zu ermöglichen (Yuan & others, 2023); es kann aber aufgrund seiner Struktur außerdem gut für Scraping verwendet werden (Gudavalli & JayaLakshmi, 2025). Da Selenium schon aus dem Jahr 2004 stammt, verfügt es über eine große Community (Almabruk et al., 2025) die Erklärungen, Anwendungsbeispiele und andere Unterstützung zur Verfügung stellt, was Selenium zunächst attraktiver erscheinen lässt. Außerdem implementiert es Selenium Core to WebDriver, welches eine Schnittstelle für viele verschiedene Programmiersprachen darstellt und eine parallele Testung mit Selenium Grid bietet (Almabruk et al., 2025). Selenium ist darüber hinaus für eine Vielzahl an Browseroptionen konzipiert, was eine flexiblere Anwendung ermöglicht. Wie bereits beschrieben, ist Python für die linguistische Verarbeitung die sinnvollste Sprache, sodass auch andere Scraper, die für Python konzipiert sind, in Frage kommen (Abodayeh et al., 2023; Bale et al., 2022). Da Selenium nicht sehr effizient, flexibel und anfängerfreundlich ist, wird im Folgenden zudem Playwright vorgestellt. Playwright ist eine von Microsoft

entwickelte öffentlich zugängliche Bibliothek und bietet im Gegensatz zu Selenium eine API, welche die drei Browser Chromium, WebKit und Firefox automatisieren kann (Almabruk et al., 2025). Playwright ist für die Programmiersprachen Java Script, Java, Typescript und Python verfügbar. Im Gegensatz zu Selenium ist es deutlich anfängerfreundlicher und übertrifft Selenium in seiner Performancegeschwindigkeit fast um das Doppelte: Playwright benötigt durchschnittlich 290,37 ms für eine Anfrage, während Selenium 536,34 ms benötigt (Melyawati et al., 2024). Durch einen *headless browser* (Bansal et al., 2023) kann Playwright Inhalte von Webseiten flexibler und schneller extrahieren (Gudavalli & JayaLakshmi, 2025) – wobei auch ein Browser *„mit Kopf"* implementiert werden kann (Persson, 2019). Bei einem Browser *„mit Kopf"* wird die zu scrapende Webseite geöffnet, sodass der Nutzer genau sehen kann, was bei dem Vorgang des Scraping passiert. Durch die Verwendung eines *headless browser* kann nicht genau verfolgt werden, wie das Programm die ausgewählte Webseite durchsucht (Bale et al., 2022). Die Verwendung eines *headless browsers* ermöglicht es, das Scraping effizienter im Hintergrund laufen zu lassen, was bei großen Datenmengen – wie auch im Fall des Scrapings der circa 7.000 Truth Social Posts von Vorteil ist. Playwright verfügt zudem über fortgeschrittenere Debugging Capabilities als Selenium. Abgesehen davon kann es durch dynamische Synchronisation im Gegensatz zu Seleniums statischem Ansatz, effizienter (schneller und mit geringeren Kosten) Tests durchführen (Almabruk et al., 2025). Playwright bietet zudem eine implementierte Methode *sleep*. Dies ermöglicht eine bessere Imitation eines menschlichen Nutzers. Im Gegensatz dazu muss diese Funktion bei Selenium selbstständig implementiert werden (Melyawati et al., 2024). Einige Webseiten werden gesperrt, wenn zu viel Aktivität auf der Seite bemerkt wird, wodurch Bots blockiert werden. Dies kann zum Beispiel durch menschenähnliches Verhalten umgangen werden (El Asikri et al., 2020). Um diese Gefahr, blockiert zu werden, wird für das Scraping der Socialmedia Posts Playwright gewählt, da es ein unauffälliges Browserverhalten aufweist.

Wie funktioniert nun das Scraping genau? Allgemein lässt sich der Prozess des Scraping laut Khder (2021) in drei Stadien unterteilen. Im ersten Stadium *Fetching Stage* sollten die gewünschten Daten der Webseite erfasst werden. Dafür wird zunächst ein Browser ausgewählt, mit dem die gewünschten Seiten aufgerufen werden können. Scraping erfordert ein tiefes Verständnis von Aufbau und Inhalt der Webseiten, um Daten richtig scrapen zu können. Vor dem Scraping sollte genau geprüft werden, welche Inhalte gescrapt werden sollen (Thole, 2022). Mit der Bibliothek Playwright beispielsweise wird dann ein DOM (Bale et al., 2022), analysiert, welches Playwright Informationen über die Webseite liefert (El Asikri et al., 2020). Alternativ oder zusätzlich kann auch mit Regex nach bestimmten Inhalten wie etwa Textinhalten oder Ähnlichem gesucht werden, wenn die Webseite die gewünschten Inhalte nicht direkt über ihre HTML-Objekte anzeigt (Yuan & others, 2023). Im Fall der Bachelorarbeit waren auf der entsprechenden Webseite alle Informationen wie etwa Datum, Uhrzeit, Plattform, Autor und Text jeweils in einem Block gespeichert (und nicht in getrennten HTML-Objekten), sodass die verschiedenen Informationen mittels Regex gesucht und in verschiedenen Spalten abgespeichert werden konnten (Abodayeh et al., 2023). Die

Extraktion der Daten erfolgt im *Extraction Stage*. Im *Transformation stage* schließlich geht es darum, die Daten in der gewünschten Form zu speichern (Khder, 2021).

2.2 POS-Tagging

Um Daten linguistisch analysieren zu können, werden diese zunächst mithilfe von Natural Language Processing (NLP) verarbeitet. Zu NLP zählen Satzsegmentierung, Tokenisierung, POS-Tagging, Lemmatisierung (Straka & Straková, 2017), NER (named entity recognition) und andere Funktionen wie Irony detection, sentiment analysis, text classification und mehr (Akbik et al., 2019; Honnibal et al., 2020; Jiang et al., 2022). Für die Bachelorarbeit sind nur die ersten vier Schritte nötig, da POS-Tags für jedes Token vergeben werden sollen, aber keine weiteren Funktionen implementiert werden. Zunächst werden die Tweets mittels Tokenizer (z.B. Twokenizer) in einzelne Sätze segmentiert, dann werden die in den Sätzen enthaltenen Wörter tokenisiert (beispielsweise wird aus don't do und not) und Satzzeichen werden von den direkt davorstehenden Wörtern getrennt (now: wird zu now :). Im nächsten Schritt können die entstandenen Tokens lemmatisiert – also auf ihre Grundform zurückgeführt werden (z.B. Token: *did*, Lemma: *do*). Im letzten Schritt können mit einem Tagger position tags vergeben werden, die Aufschluss über die Wortart und Position des Tokens im Satz geben. POS-Tags können dabei auch Satzzeichen oder Emojis sein. Außerdem muss ein Parser (z.B. Tweeboparser) durch die Texte iterieren und diese parsen. Unter POS-Tagging versteht man meistens nicht nur das Vergeben der position tags, sondern den gesamten eben beschriebenen Prozess bestehend aus den vorherigen Schritten (Segmentieren, Tokenisieren, Lemmatisieren), da in der Regel rohe, unverarbeitete Texte verwendet werden. Im Folgenden werde ich zunächst auf Datensätze und deren Tagsets eingehen, die die Grundlage für POS-Tagging bilden. Anschließend werde ich verschiedene Tagger mit deren Vor- und Nachteilen vorstellen.

Datensätze für POS-Tagging

Die jeweiligen Tagging-Modelle wurden auf großen Trainingsdaten trainiert, um möglichst gute Ergebnisse zu erzielen. Für die deutsche Sprache gibt es zum Beispiel das TIGER-Korpus, auf dem Tagger für Deutsch häufig trainiert werden (Brants et al., 2004). Für verschiedenste Sprachen stellt Universal Dependencies zahlreiche Datensätze zur Verfügung (Nivre et al., 2016). Fürs Englische gibt es das Brownkorpus (Francis, 1964), Korpora der Universal Dependencies (zum Beispiel English Web Treebank (EWT) bestehend aus Webtexten wie weblogs, reviews, newsgroups) und Penn Treebank (Marcus et al., 1993), die gut fürs Tagging englischer Texte geeignet sind. Um twitter-spezifische Merkmale abdecken zu können, wurde ein neuer Datensatz, der zunächst aus 1.827 Tweets des 27. Oktober 2010 besteht (namens OCT27), manuell getaggt (Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters). Dieser wurde um einen neuen Datensatz bestehend aus 547 Tweets (namens DAILY547) ergänzt. Dafür wurde jeden Tag zwischen dem 1.1.2011 und dem 30.06.2012 ein zufälliger Tweet von Twitter ausgewählt und gesammelt. Zudem wurde ein neues Twittertagset verwendet, das im

nächsten Absatz zu Tagsets beschrieben wird (Gimpel et al., 2011). Aus den gesammelten Daten von Owoputi et al. (2013) und Gimpel et al. (2011) wurden für Tweebank V1, eine neue Datenbank für Tweets, 929 Tweets mit insgesamt 12,318 Token entnommen (Kong et al., 2014). Aufgrund fehlender Annotationsguidelines und nur wenig Zeit, in der der Datensatz annotiert werden musste, weißt er nun an einigen Stellen grobe Fehler auf. Parallel entwickelte Foster et al. (2011) einen Datensatz, der sich am Tagset der Penn Treebank orientiert, aus 7.630 Tokens besteht und in Stanford Dependencies umgewandelt werden kann. Aufgrund dessen besserer Performance wurde der Datensatz von Kong et al. (2014) technisch neu annotiert. In der letzten Veränderung wird nun Tweebank V2 erzeugt. Dieses ist viermal größer als Tweebank V1 und besteht aus 55.607 Tokens, die den Universal Dependency Richtlinien folgen, um mehr Klarheit im Umgang mit informeller Sprache zu schaffen (Liu et al., 2018). Für den Umgang mit informeller Sprache gab es bei Tweebank V1 keine klaren Regelungen, sodass nicht konsistent annotiert wurde. Nun wird Tweebank V2 mit POS-Tags und Universal Dependencies ausgestattet. Tweebank V1 ist nicht sehr konsistent in der Annotation, sodass bei Tweebank V2 alle alten Tweets neu annotiert wurden (ohne MWEs), um konsistent mit Universal Dependencies und EWT (English Web Treebank) konform zu sein (Liu et al., 2018). Wenn irreguläre Ausdrücke wie „idc“ (I don’t care) oder „mfw“(my face when) auftauchen, werden diese nach UD Konventionen getaggt, sodass ersteres als VERB und zweiteres als NOUN getaggt wird. Emoticons werden mit SYM getaggt und die meisten anderen unsyntaktischen Ausdrücke (die nicht zur Syntax des Satzes gehören) wie oft auch Hastags oder URL werden mit X (other) markiert. Erwähnungen (@) werden als vokative Ausdrücke behandelt und mit PROPN getaggt (Liu et al., 2018). Tweebank V2 besteht aus 840 Tweets von TweebankV1, 210 Tweets von DAILY547 und 2.500 neuen Tweets von Februar 2016 bis Juli 2016 (Liu et al., 2018). Mit speziellen Guidelines und vorheriger Schulung durch ein Tutorial wurden die alten Tweets (Tweebank V1 und DAILY547) innerhalb eines Tages neu annotiert. Danach wurden alle neu dazu gekommenen Tweets mithilfe der Guidelines in Abstimmung mit den Bestimmungen von Universal Dependencies sorgfältig annotiert. Tweebank V2 enthält insgesamt 55.607 Tokens und stellt nun einen umfangreichen Datensatz nach dem Vorbild der Penn Treebank dar. Der Datensatz ist für das Tagging von Tweets geeignet (Liu et al., 2018).

Tagsets

Im Folgenden wird eine Auswahl möglicher Tagsets für das Tagging vorgestellt, auch wenn es darüber hinaus viele weitere Tagsets gibt. Eines der ersten Tagsets zur Analyse von Satzstrukturen war das Tagset des Brownkorpus, dem ersten umfangreichen englischen vollständig annotierten Korpus, mit etwa 154 verschiedenen Tags, die genaue Informationen wie das Tempus der Verben oder nähere Bestimmungen von Modalpartikeln oder Nomen bereitstellten. Es vergibt Tags zum Beispiel für have (HV), do (DO, DOD, DOZ) und be (BEM, BER, BEDZ). Allerdings war das Tagset aufgrund seines Umfangs sehr komplex und fehleranfällig, da ein Satz manchmal keine klare Lesart aufweist nicht jedes Wort klar disambiguiert werden kann (Marcus et al., 1993). Daraufhin wurde das Penn Treebank Tagset entwickelt. Dieses

basiert auf Penn Treebank, einem umfangreichen, vollständig getaggten Korpus bestehend aus über 4.5 Millionen Wörtern des amerikanischen Englisch. Das Tagset weist 28 verschiedene Tags auf und ist im Verhältnis zum Tagset des Brownkorpus stark vereinfacht. Es fasst die Tags für have (HV), do (DO, DOD, DOZ) und be (BEM, BER, BEDZ) zu allgemeineren Tags aufgeteilt in die verschiedenen Zeitformen VB, VBZ, VBD, VBG und VBN zusammen. Das kleinere Tagset bietet keine so umfangreichen grammatikalischen Informationen wie das des Brownkorpus, wodurch es jedoch deutlich weniger fehleranfällig ist. Dies ist für eine genaue Analyse der Grammatik und anderer sprachlicher Vorkommnisse Ausschlag gebend (Marcus et al., 1993). Neben dem Penn Treebank Tagset (auch mit pos abgekürzt) gibt es außerdem das Tagset der Universal Dependencies (als upos abgekürzt) mit 17 verschiedenen Tags. Universal Dependencies folgt den Bestrebungen, für möglichst viele Sprachen sowohl Datensätze als auch ein einheitliches Tagset zur Verfügung zu stellen, sodass Nutzer auf der ganzen Welt auf ein ähnliches Verständnis der Tags zurückgreifen können (Nivre et al., 2016). Aufgrund dieser einheitlichen Regelung können verschiedene Sprachen besser analysiert, verstanden und verglichen werden. Auf der anderen Seite werden viele Satzarten nur grundlegend bestimmt. Wo Penn Treebank beispielsweise differenziert, ob es sich um Singular (NN) oder Plural (NNS) eines Wortes handelt, vergibt Universal Dependencies lediglich das Tag NOUN. Pronomen beispielsweise werden bei Penn Treebank in Personal- (PRP), Possessiv (PRP$), Wh- (WP) und Possessive Wh-Pronomen (WP$) unterteilt, während Universal Dependencies nur ein einziges Tag PRON vergibt (Nivre et al., 2016). Beide Tagsets sind deutlich kompakter als das Tagset des Brownkorpus und stellen zwei gute Optionen für POS-Tagging dar (Marcus et al., 1993; Nivre et al., 2016). Neben diesen beiden gängigen Tagsets gibt es ein speziell für Tweets entwickeltes Tagset (Gimpel et al., 2011). Dieses berücksichtigt verschiedene tweet-spezifische Phänomene wie etwa Erwähnungen mit @, Hashtags, Emojis und Abkürzungen bzw. dialektalen Ausdrücken. Herkömmliche Tagger wurden auf Treebanks wie etwa die Penn Treebank trainiert, welche längere und grammatikalisch komplexe Texte enthält. Auf Twitter wurden Tweets auf 140 Char begrenzt, enthalten Slang (runnin‘), Abkürzungen (lmao) und alternative Schreibweisen von Wörtern (mommma). Daher wurde ein neues Tagset entwickelt, das auf 1,827 annotierten Tweets basiert. Entwickelt wurden 25 Tags; T steht beispielsweise für verb particle „out, off, Up, UP“, D für determiner „the, teh, its, it’s, V für verb incl. Copula, Auxiliaries “might, gonna, ought, is, eats”, $ für numeral “2010, four, 9:30” und twitter-spezifisch # für Hashtags, die ein Thema des Tweets anzeigen “#acl”, @ für Erwähnungen, U für URL oder Email-Adressen http://.., E für Emojis „:-), :b, (:, <3, o__O“ (Gimpel et al., 2011). Neben den eben vorgestellten Tagsets gibt es eine Vielzahl weiterer Tagsets für die verschiedensten Sprachen (Hardie, 2023).

5 Modelle

Wenn es nun an NLP – und insbesondere POS-Tagging geht, so gibt es eine Vielzahl verschiedener Tokeniser, Tagger und Parser, die mit verschiedenen Methoden verschiedene Bereiche des NLP für verschiedenste Sprachen anbieten. Für das Deutsche gibt es zum Beispiel EmpiriST (Remus et al., 2016) oder HanTa (Wartena, 2023), welcher zudem mit Niederländisch und Englisch umgehen kann. Für die

Bachelorarbeit ist ein vortrainiertes Modell nötig, das auf englischen Texten trainiert wurde. Außerdem sollte es mit Tweets umgehen können – also kürzeren, meist informellen Textbeiträgen. Allerdings wurden die meisten Tagger auf Zeitungsartikeln, längeren Texten oder Blogbeiträgen trainiert. Nur wenige Tagger wurden speziell auf Tweets trainiert. Da diese anders als normalsprachliche Texten strukturiert sind, können herkömmliche Tagger meist keine guten Ergebnisse bei Tweets erreichen (Gimpel et al., 2011). Im Folgenden werde ich fünf State-of-the-Art-Tagger vorstellen. Einer der ersten Tagger für normalsprachliche Texte war **UDPipe** (Straka & Straková, 2017). Der Tagger bringt eine modifizierbare Pipeline für Satzsegmentierung, POS-Tagging, Lemmatisierung und Dependency parsing für 50 verschiedene Sprachen mit. Das Modell selbst ist in C++ geschrieben, was für Python-nutzer ungeeignet ist. Allerdings bietet eine Pipeline Anbindungen für Python, C++, Perl, Java and J#. Fürs Lemmatisieren wird ein F1 Score von 96.10 erreicht, für UPOS 93.50 und für XPOS 92.88 (Straka & Straková, 2017). Diese Ergebnisse wurden mittlerweile durch andere Tagger übertroffen. **Flair** ist ein weiterer Tagger, der verschiedene Modelle für normalsprachliche Texte bietet (Akbik et al., 2019). Bei dessen Konzeption standen einfache Anwendung und Nutzerfreundlichkeit durch eine gute Dokumentation im Vordergrund. Jeder Satz wird als Sentence Objekt gespeichert. Für die einzelnen Token des Satzes gibt es Felder für POS- oder NER-Tags und Embeddings, die auf GloVe oder einer Vielzahl anderer von Flair zur Verfügung gestellten Embeddings (ELMo, BERT, Pooled Flair Embeddings) basieren. Flair stellt außerdem Vektorrepräsentationen für Wörter und ganze Dokumente zur Verfügung. Alle Wortembeddings werden zu einem Dokumentembedding abgeleitet, welches dann an ein LSTM gefüttert wird und daraufhin eine Vektorrepräsentation für die Dokumente liefert. Außerdem gibt es viele weitere Features, mit denen eigene Modelle trainiert werden können (Akbik et al., 2019). Allerdings können über Sequencetagger auch vortrainierte Modelle verwendet werden. Es gibt nicht nur monolinguale, sondern auch multilinguale POS-Modelle. Von den vortrainierten monolingualen Modellen sind vier Modelle für englisches POS Tagging geeignet, von denen zwei als fast-Variante verwendet werden können. Die schnellen Varianten haben statt 2048 hidden states nur 1024 und können auf einer CPU statt der GPU laufen (Akbik et al., 2019). Das erste Modell flair/pos-english hat einen F1 Score von 98,18 und nutzt das Penn Treebank Tagset. Die zweite Version des Modells flair/pos-english-fast ist die schnelle Version, die 98,10 erreicht. Auch für das Tagset der universal Dependencies bietet Flair ein normales Modell, welches im F1 Score 98,6 erreicht und eine schnelle Version, die 98,47 erreicht (Akbik et al., 2019). Flair wurde allerdings nicht speziell auf Tweets trainiert, sodass ungewiss ist, ob die State-of-the-Art Ergebnisse bei Tweets beibehalten werden können. Flair hat keinen Lemmatizierer, der das Lemma der jeweiligen Token herausfindet und speichert. Allerdings kann dafür SpaCys Lemmatisierer genutzt werden. Außerdem verwendet Flair Spacys Tokenizer (Jiang et al., 2022), da es keinen eigenen implementiert. **SpaCy** (Honnibal et al., 2020) ist ein sehr etablierter Tagger, der sowohl lemmatisieren als auch taggen kann. Das SpaCy-Transformer Framework ermöglicht es, vortrainierte Repräsentationen auf Transformers basierender Sprache und Spacys eigenen NLP Modellen über Transformers von Hugging Face (Jiang et al., 2022) zu verwenden. Das Lemmatisieren ist regelbasiert und nutzt ein Wörterbuch, um die jeweiligen Wörter und deren POS-

Tags zu überprüfen (Jiang et al., 2022). Die Webseite von SpaCy beschreibt alle Anwendungsmöglichkeiten und ist sehr gut, was das Modell sehr benutzerfreundlich macht.

Stanza ist der state-of-the-art Stanford Tagger, der in einer früheren Version auch als StandfordNLP bekannt war (Qi et al., 2020). Der Tagger wurde auf 112 Datensätzen trainiert und liefert für 66 (nach dem neuesten Update 2024 sogar 70) verschiedene Sprachen Modelle. Er ist zudem ebenso wie Flair, SpaCy und UDPipe open source und daher für jeden zugänglich. Im Unterschied zu UDPipe und SpaCy ist Stanza vollkommen neuronal trainiert - ebenso wie Flair (Qi et al., 2020) . Stanza besteht aus zwei Komponenten. Erstens besteht es aus einer neuronalen Pipeline, die Tokenisierung, Lemmatisierung, Dependency Parsing, POS Tagging, NER und andere Funktionen liefert. Zweitens besteht es aus einer Python-Schnittstelle für die Java Stanford CoreNLP Software. Der Tagger kann mit rohen (unvorverarbeiteten) Texten umgehen und diese vorverarbeiten und tokenisieren, bevor sie weiter verarbeitet werden. Das Modell versucht vorherzusagen, ob es sich bei den jeweiligen Zeichen um das Ende eines Tokens, das Ende eines Satzes oder das Ende einer MWE (engl. Multi-word-unit) handelt. Die einzelnen Tokens eines Satzes werden anschließend mit POS-Tags versehen. Dafür werden POS und UFeats mithilfe eines Bi-LSTM vorhergesagt. Neben den Universal Pos-Tags stellt Stanza auch ein Treebank-spezifisches Tagset xpos zur Verfügung. Für Tweebank V2 sind das zum Beispiel ADD und NFP, die tweet-spezifische URLs und Emojis abdeckt. Lemmatisiert wird über eine Kombination aus einem wörterbuch-basierten und einem neuronalen seq2seq Lemmatisierer. Dependency Parsing gelingt mithilfe eines erweiterten Bi-LSTM-basierten Biaffine Dependency Parser von Dozat und Manning (2016). Trainiert man nun die neuronale Pipeline, so werden alle Ergebnisse in einem Dokument gespeichert. Die neuen Annotationen werden überdies in Sätzen, Wörtern und Tokens gespeichert (Qi et al., 2020). Bei der Evaluation verschiedener Datensätze im Vergleich zu UDPipe und SpaCy erreicht Stanza die besten Ergebnisse. Beim Datensatz EWT erreicht Stanza bei Tokens 99.01%, bei Sätzen 81.13%, bei Lemmas 97.21%, bei UPOS 95.40% und beim spezifischen Tagset XPOS 95.12%. Stanza wurde nicht auf Tweets trainiert und erreicht bei Twittertexten daher nur circa 73% (Gui et al., 2017). Andere Tagger schneiden allerdings ähnlich schlecht oder schlechter ab (Gui et al., 2017; Owoputi et al., 2013). Insgesamt ist Stanza ein wettbewerbsfähiger Tagger, der sehr viel Flexibilität aufgrund der vielen verschiedenen Kombinationsmöglichkeiten und der 70 verschiedenen Sprachen bietet. Allerdings ist die Recheneffizienz nicht so gut wie bei anderen Modellen, sodass Stanza lange für NLP-Anwendungen braucht. Angesichts der guten Perfomance, Anwenderfreundlichkeit und Flexibilität kann darüber hinweggesehen werden.

TweebankNLP ist ein neuerer Bertweet-basierter Tagger, der auf Tweebank V2 trainiert wurde. Der Tagger ist eine Modifikation basierend auf der Python Bibliothek Transformers. Außerdem wird von den selben Autoren eine NLP-Bert-basierte Twitter-Stanza-Pipeline veröffentlicht, die auf Stanzas Tagger basiert. Allerdings werden die xpos (sprachspezifisches Tagset von Stanza) außen vor gelassen, da Tweebank V2 nur upos aufweist (Jiang et al., 2022). Diese wurde auf Tweebank V2 trainiert und schneidet im Vergleich zu anderen Modellen wie Flair oder SpaCy deutlich besser ab (Jiang et al., 2022).

TweebankNLP selbst ist ein transformer-basiertes Modell, welches vor allem mit Fokus auf NER trainiert wurde. Aber auch bei POS Tagging und Dependency Parsing schneidet es gut ab. Im Paper zu Tweebank-NLP wird zudem beschrieben, dass Tweebank V2 mit Tags für NER annotiert wurde. Außerdem wurden verschiedenste Bereiche der State-of-the-art Tagger miteinander verglichen. Alle Tokenizer wurden auf Tweebank V2 und UD-English-EWT neu trainiert. Ebenso wurden die Performance Lemmatisierer, POS-Tagger und Dependency Parser auf Tweebank V2 und EWT trainiert und miteinander verglichen. Stanzas Tokenizer mit Tweebank V2 schneidet beispielsweise mit 98.64% a besten ab. Beim POS-Tagging erreicht Stanza ohne Tweebank V2 90.6% und mit TB V2 93.20%. Das BERTweet-Modell von Huggingface in Kombination mit Tweebank V2 und EWT schneidet mit 95.38% am besten ab (Jiang et al., 2022).

2.3 CADS

Der Begriff CADS – corpus-assisted discourse studies, geprägt durch Baker und Mautner (Baker, 2006; Baker et al., 2008; Mautner, 2009) basiert ursprünglich auf CDA, der kritischen Diskursanalyse (Fairclough, 2015). Die kritische Diskursanalyse betrachtet Sprache als Diskurs und soziale Interaktion. Dabei untersucht sie das Verhältnis zwischen Sprache und Ideologie und untersucht, welche Bedeutung hinter den gewählten Worten steckt. Jedem Vortrag, jeder Nachricht und jedem Tweet geht die Entscheidung des Autors voraus, welche Worte verwendet werden sollen. Dies ist meistens ein unbewusster Vorgang - bei öffentlichen Reden oder Posts von Politikern ist dieser Vorgang wahrscheinlich bewusster gewählt als bei alltäglichen Unterhaltungen oder Nachrichten. Denn dadurch, dass die Worte oder Tweets eines Politikers wie beispielsweise Trump deutlich mehr Leute lesen werden als etwa meine Posts, erhält die Bedeutung der dahinter liegenden Worte eine weitaus größere Bedeutung und Tragweite. Ob bestimmte Sätze nun bewusst oder unbewusst geschrieben werden, ändert nichts an der Tatsache, dass entsprechende Sätze dennoch Aufschluss über das zugrunde liegende Gedankengut der Autoren geben und eine beachtliche Wirkung auf die Empfänger der jeweiligen Nachrichten verursachen (Wodak, 2015).

Thematisch werden häufig Narrative zu globalen Problemen und Herausforderungen wie Klimawandel (Grundmann & Krishnamurthy, 2010; Wang & Huan, 2023) oder die Analyse politischer Positionen (Baker & McEnery, 2005; Wodak, 2015) zur Analyse ausgewählt. Allerdings werden Korpora bei CDA meist genau unter dem Blickwinkel von bestimmten vorher ausgewählten Ideologien oder Narrativen untersucht, sodass es zu „Cherry Picking“ (Rosinenpickerei) kommt (Baker & Levon, 2015). Aufgrund der großen Datenmenge der jeweiligen Korpora müssen Forscher sich auf bestimmte Themen beschränken und wählen auf diese Weise selektiv aus, welche Vorkommnisse ihre Argumentation unterstützen und welche nicht. Darum wird oft der Vorwurf der Subjektivität gegenüber CADS geäußert. Wenn nur qualitative Forschung betrieben wird, müssen die Ergebnisse subjektiv gewählt werden. In Kombination mit statistischen Korpusmethoden können diese Vorwürfe möglicherweise behoben werden. Ziel wäre es, auf der einen Seite einen Korpus-gesteuerten Ansatz zu wählen (corpus-driven), bei dem man das Korpus aus einer naiven Perspektive heraus analysiert und vor allem Frequenzlisten und Kookurrenzen betrachtet.

Auf Basis dieser Erkenntnisse können neue Zusammenhänge und Auffälligkeiten der Korpora analysiert und bewertet werden (Heinrich & Evert, 2020). Auf der anderen Seite kann man auch den korpusbasierten Ansatz (corpus-based) wählen, bei dem man bereits mit bestimmten Hypothesen startet und diese dann mithilfe der Korpusanalyse bestätigen oder widerlegen kann. Gerade hier ist wichtig, möglichst neutral und objektiv an die Analyse heranzugehen, damit kein falscher Eindruck entsteht und beispielsweise nur die 10 Fälle, in denen ein Motiv vorkommt, beschrieben werden anstatt der unzähligen anderen, in denen das nicht der Fall ist (Zih et al., 2022). Am besten sollten diese beiden Methoden vermutlich kombiniert und rekursiv ausgeführt werden, um transparente Ergebnisse der CADS zu ermöglichen (Zih et al., 2022). Auch Baker schlägt eine Kombination aus quantitativen und qualitativen Ansätzen zusammen mit soziokulturellem Hintergrund vor (Baker & Levon, 2015). Bei CADS sollte mit der Untersuchung der Keywords und Kollokationen eines Korpus begonnen werden (Baker, 2006; Baker et al., 2008). Schlüsselwörter (Keywords) sind Worteinheiten – meistens Lemmata (Heinrich & Evert, 2020) – mit einer signifikant höheren Häufigkeit in einem Zielkorpus im Vergleich zu einem Referenzkorpus. Die Schlüsselwörter weisen außerdem auf bestimmte Themen, Schlüsselbegriffe und dazugehörige Narrative hin. Um diese systematisch finden zu können, sollte mit initialen Begriffen begonnen werden. Heinrich und Evert (2020) stellen dafür den Begriff *Diskursem* (engl. „Discourseme“) vor. Diskurseme stellen eine Menge lexikalischer Worteinheiten mit bestimmter diskursiver Funktion (COVID-19, Grippe, Erkältung, harmlos) dar. Diese können auch aus mehr als einem Wort bestehen und somit auch Eigennamen (Named Entities), Zeitausdrücke, fixe Idiome oder Phrasen beinhalten. Die Diskurseme sollten zudem POS-disambiguiert sein. Diskurseme fangen oft nur einen Teil-Aspekt der diskursiven Position ein und können Teil mehrerer diskursiver Positionen sein. Themendiskurseme stellen dabei den Knotenpunkt der Kollokationsanalysen dar und sind Diskurseme, die a priori definiert werden (Heinrich & Evert, 2020). Begonnen werden sollte also mit der Definition der Themendiskurseme, auf Grundlage derer mit der Konkordanzanalyse begonnen werden kann. Im Laufe des Prozesses können die Diskurseme der Analyse sich erweitern oder verschieben, abhängig von den gefundenen Ergebnissen. Die so gewonnenen Keywords und Kollokationen wurden durch quantitative Methoden, oder auch *distant reading* (Heinrich & Evert, 2020) gefunden und können nun mithilfe qualitativer Methoden oder *close reading* (Heinrich & Evert, 2020), also bei näherer Betrachtung der Wörter und deren Kontext genauer untersucht werden. Dabei sind die verwendeten Algorithmen (association measures und parameter settings) zur Berechnung von Keywords oder Konkordanzen und Kollokationen ausschlaggebend. Gerade die bewusste Wahl der Algorithmen gestaltet derartige Analysen aussagekräftig und repräsentativ, da entsprechende Analysen so besser nachvollzogen werden können. Allerdings gibt es dabei keinen bestimmten Algorithmus, mit dem Konkordanzen berechnet werden sollten (Evert, 2022; Evert et al., 2017; Stubbs, 1995), sodass mitunter an Korpus-Analyse-Tools gearbeitet wird, die verschiedene Algorithmen wie etwa log-likely-hood-filtered odds-ratio oder conservative log ratio (Evert, 2022) zur Auswahl stellen. Die eben genannten Algorithmen kombinieren statistische Signifikanz mit Effektgröße (effect size measures) wie Evert (2022) schreibt, was für den Beginn einer Analyse ratsam ist. Zunächst werden Kollokationen und

Schlüsselwörter gruppiert, woraufhin eine Interpretation der Ergebnisse erfolgt. Anschließend können die gefundenen Keywords und Kollokationen interpretiert werden. Auf dieser sogenannten „Meso-Ebene" der Diskursanalyse (Fairclough, 2015) befindet sich die Brücke zwischen linguistischen (und korpuslinguistischen) und diskursiven Mustern. Auf der Meso-Ebene erfolgt die Interpretation quantitativer Ergebnisse und bildet somit den Übergang zur qualitativen Analyse, die beispielsweise den Kontext der gefundenen Schlüsselwörter mehr einbezieht. Man könnte ab diesem Zeitpunkt von einer Mixed Methods Discourse Analysis (MMDA) sprechen (Evert & Heinrich, 2019), unter der genau die Kombination dieser Muster verstanden wird. Gemischter Methodiken bedient sich zum Beispiel Pérez, die quantitative Methoden mit qualitativen diskurs-historischen Ansätzen kombiniert (Pérez, 2024). Zu einer Analyse wie CADS gehören klassischerweise folgende Features: Suchanfragen mit dem Corpus Query Processor (CQP) - also Suchanfragen über CQP (Evert & The CWB Development Team, 2022), ein Tool, das auch bei CQPweb oder CWB (Evert & Hardie, 2011) implementiert ist, Bildung von Konkordanzen, Query breakdown, Einbezug der Metadaten, Erstellung mehrerer Subkorpora, Kollokations- und Keywordanalyse (Heinrich & Evert, 2020). Außerdem sollte CADS nach den Prinzipien Sampling (Auswahl zufälliger Treffer), Filtering, Cleaning (das Bereinigen der Daten wenn beispielsweise andere Bedeutungen eines Wortes erscheinen, die nicht gesucht werden), Sorting, Ranking und Grouping vorgenommen werden (Piperski et al., 2025).

2.4 Konkordanz Tools zur Korpusanalyse

Um Korpusanalysen und Studien wie CADS zu unterstützen, gibt es verschiedene Tools, die Funktionen wie Frequenz-, Keyword- und Kollokationsanalysen mitliefern. Dabei bedienen sie sich verschiedener Algorithmen. Nicht jedes Tool bietet jeden Algorithmus zur Berechnung von Konkordanzen oder Keywords, sodass der Nutzer auf die jeweils verfügbaren Algorithmen zurückgreifen muss. Nicht alle heuristischen Maßeinheiten beinhalten statistische Grundlagen wie etwa SimpleMaths von SketchEngine (Evert, 2022; Kilgarriff, 2009). Im Folgenden werde ich zunächst verschiedene Konkordanztools vorstellen und anschließend eine Lösung für die Verwendung verschiedener Algorithmen anbieten. Es gibt zahlreiche Konkordanztools (concordancing tools), die sich in zwei Gruppen einteilen lassen. Zum einen gibt es offline nutzbare Konkordanztools (Heinrich & Evert, 2020) wie etwa AntConc, WordSmith und LancsBox. Zum anderen gibt es Web basierte Konkordanzer wie SketchEngine und BNCweb oder CQPweb (Weisser, 2020).

AntConc ist ein Korpus-Analyse-Tool von Laurence Anthony der Universität Waseda, Japan (Anthony, 2005, 2014). Ursprünglich wurde es fürs Unterrichten und Lernen einer Schule in Osaka entwickelt. Nach einem Update implementiert AntConc KWIC Konkordanzen (Concordance Tool), Diagramme zur Verteilung von Suchanfragen (Concordance Plot Tool), Ansicht der Originaldaten (File View Tool), Word Cluster und Lexikalische Anhäufungen (Clusters/N-Grams), Wordlisten (Word List) und Schlüsselwortlisten (Keyword List). Auch RegEx und Wildcards sind für Suchen implementiert (Zih et al., 2022). Das

erste genannte Tool zeigt Suchergebnisse in KWIC an, sodass auch der Kontext der Suchanfrage genauer beleuchtet werden kann. Mithilfe des Buttons „Advanced Search“ können auch komplexere Anfragen gestellt werden. In einem „Bar code“-Format werden Konkordanzen angezeigt. Die Länge des Textes wird als Breite des Balkens dargestellt, sodass gesehen werden kann, an welchen Stellen sich die meisten Treffer befinden. Mit dem Tool zur Ansicht der Originaldateien können einzelne Dateien vollständig (ohne KWIC) angezeigt werden, da der Kontext nicht beschränkt wird. Auch hier können Suchanfragen inclusive RegEx gestellt werden. Im Clustertool können bestimmte Wortmuster und Gruppen angezeigt werden. Im Wesentlichen fasst es die Ergebnisse des Konkordanztools und des Plotting Tools zusammen. Im N-Gramms Tool können Worteinheiten bestehend aus mehreren Wörtern gesucht werden. Nach Kollokaten kann mit dem Kollokat-tool gesucht werden. Die Ergebnisse werden sortiert nach Häufigkeit oder anderen Parametern angezeigt. Die verwendeten Algorithmen beruhen auf T-Score und MI basierend auf den Beschreibungen von „Collocations and Semantic Profiles, Functions of Language 2, 1“ von Stubbs (1995). Im Word List Tool werden Worte nach der Häufigkeit ihres Auftretens im Korpus sortiert, während bei der Keyword List die häufigsten Wörter des Zielkorpus im Vergleich zu einem ausgewählten Referenzkorpus angezeigt werden. Berechnet werden diese unter anderem mit dem Log Likelihood Algorithmus (Anthony, 2014).

SketchEngine (Kilgarriff et al., 2014) hat keine statistische Grundlegung (SimpleMaths: Kilgarriff, 2009; aus Evert, 2022). SketchEngine ist nicht frei verfügbar; nach einer dreißigtägigen Testphase ist für die weitere Nutzung eine kostenpflichtige Lizenz erforderlich. Allerdings bietet es für verschiedene Sprachen, Korpora und Korpustypen Anbindungen (Kilgarriff & Kosem, 2012). Konkordanzen werden im KWIC (Key Word in Context) Format angezeigt, bei dem jedes über eine Suche gefundene Wort mit Kontext links und rechts angegeben wurde, was *close reading* ermöglicht. Bei der Analyse von Konkordanzen werden auffällige Konkordanzen gruppiert, anschließend sortiert, um Ergebnisse beispielsweise nach dem ersten Wort links oder rechts neben dem gesuchten Wort oder Lemma zu filtern. Mithilfe von Sampling können zufällige Textstücke des Korpus ausgewählt werden, um einen besseren Überblick über das Korpus zu bekommen. Betrachtet man beispielsweise nur die ersten 50 Einträge, so könnte man völlig falsche Schlüsse ziehen. Durch die zufällige Sortierung (Sampling) kann Derartiges vermieden werden, was eine qualitativ hochwertige Analyse unterstützt. Durch Filtern können bestimmte Muster ausgewählt werden, die näher untersucht werden sollen. Dies kann durch die Einteilung in sogenannte Subkorpora unterstützt werden. Mithilfe von Häufigkeitsanalysen kann das Vorkommen bestimmter Wörter beispielsweise mit Subkorpora zu verschiedenen Zeitabschnitten oder den Gebrauch bestimmter Ausdrücke in anderen Korpora verglichen werden. Außerdem gibt es Aufschlüsse über den sprachlichen Stil des untersuchten Korpus. Neben den eben beschriebenen Funktionen implementiert SketchEngine zudem Thesaurus mit; ein Feature, bei dem die benachbarten Wörter (nearest neighbours) des gesuchten Ausdrucks aufgelistet werden. Es handelt sich dabei um Wörter, die die meisten Kollokate mit ihrem Knotenwort teilen (Kilgarriff

& Kosem, 2012). Die Ergebnisse der Konkordanzanalysen können bei SketchEngine außerdem in xml exportiert werden.

BNCWeb (Lehmann & Schneider, 2000) und **CQPweb** (Hardie, 2012; Hoffmann & Evert, 2005) stellen zwei über das Internet zugängliche Tools. BNCweb wurde ursprünglich für das BNC Korpus geschrieben und ist nur darauf anwendbar, sodass andere Korpora damit nicht analysiert werden können. Allerdings ist das Tool sehr anwenderfreundlich und übersichtlich, sodass an einer öffentlich nutzbaren Version gearbeitet wird, mit der verschiedene Korpora analysiert werden können (Hardie, 2012). CQPweb ist eine solche Kombination aus BNCweb und CQP – der effizienten und flexiblen Sprache für Suchanfragen, die bei der IMS Corpus Workbench (CWB) implementiert ist. BNCweb basiert auf einer SARA Server Software. Ergebnisse werden entweder in KWIC oder als Liste ganzer Sätze angezeigt. Außerdem können über Links weitere Kontextinformationen und weitere Sprecherinformationen angezeigt werden, wenn diese vorhanden sind. Daneben bietet BNCweb auch weitere Funktionen zur Analyse des Korpus wie etwa das Anzeigen sortierter Suchergebnisse, Kollokationen, Häufigkeitsverteilungen, und mehr. Über den Suchverlauf können Ergebnisse gespeichert und für weitere Analysen verwendet werden (Hoffmann & Evert, 2005). Allerdings weist BNCweb auch einige Schwächen auf: So können zum Beispiel keine Suchanfragen mit grammatikalischen Elementen (Lemma, POS-Tags) generiert werden. Außerdem kann BNCweb nur mit dem Korpus BNC arbeiten (Hoffmann & Evert, 2005). Aus diesem Grund wurde BNCweb mit CWB fusioniert. Letzteres ermöglicht die Suche mit der Sprache CQP und erleichtert damit die Suche nach grammatikalischen Elementen, Strukturattributen und Metadaten. CWB bietet ebenfalls eine Ansicht der Ergebnisse in KWIC und fokussiert sich auf die Analyse von Lemmatisierung und morphologische Merkmale. Auch in CWB können Frequenzlisten berechnet und Ergebnisse gespeichert/heruntergeladen werden. Leider ist CWB nicht so anwenderfreundlich wie BNCweb, da CQP zunächst schwer zu verstehen ist, sodass insbesondere Anfänger und Linguisten nicht auf dieses Tool zurück greifen. Zudem wird kein graphisches Interface implementiert, weshalb Befehle im Terminal ausgeführt werden müssen (Hoffmann and Evert, 2005). Das viel später entstandene Tool CQPweb (Hardie, 2012) kombiniert die Stärken der beiden Tools, sodass beispielsweise sowohl simple queries als auch CQP möglich sind. In CQPweb können im Unterschied zu BNCweb verschiedene Korpora (mit verschiedenen Tagsets) untersucht werden. Zudem werden Funktionen wie die Bildung von Konkordanzen, Kollokationen oder Keywords unterstützt. Auch Frequenzlisten und Verteilungstabellen sind Teil von CQPweb. Ergebnisse werden in KWIC in zufälliger Reihenfolge gezeigt, um einen guten Eindruck der Daten zu erhalten. Außerdem sind Metadaten über einen Klick auf entsprechende Zeilen der Konkordanzen in KWIC einsehbar. Auch ein Frequency Breakdown der Schlüsselwörter, das Bilden von Subkorpora und die Verwendung verschiedener Algorithmen sind möglich. Insgesamt ist CQPweb wesentlich anwenderfreundlicher als CWB und flexibler als BNCweb.

Cwb-ccc ist ein Python Wrapper, der in Kombination mit der IMS Korpusworkbench (CWB) funktioniert und ohne Web- oder App-interface betrieben wird. Verwendet man es beispielsweise in Jupyter-

Notebooks, so stört das jedoch nicht. Zudem werden alle wichtigen Funktionen wie Kollokations- und Keywordanalyse, Bildung von Subkorpora, Verwendung verschiedener Algorithmen und Frequency Breakdown implementiert (Heinrich, 2025).

Neben den eben genannten Tools gibt es eine Vielzahl weiterer Konkordanzer (Scott, 2024; XARA: Burnard & Dodd, 2003; Gaizauskas et al., 2003) oder andere, siehe: (Weisser, 2020), die ebenfalls mehr oder weniger ähnliche Funktionen implementieren. Da die Meisten allerdings nicht immer auf statistischen Grundlegungen basieren und nicht so viele verschiedene Algorithmen implementieren, wurde **FlexiConc** (Dykes et al., 2025a) entwickelt. Da es aufgrund der bereits unüberschaubaren Zahl keines weiteren Konkordanzers bedarf, ist FlexiConc ein Tool, das mit verschiedenen Konkordanztools zusammenarbeitet. Bisher kann FlexiConc Queries von CWB, CQPweb, SketchEngine, CLiC (Mahlberg et al., 2016) und WMatrix (Rayson, 2009) importieren. Wo den einzelnen Tools teilweise die statistisch fundierten Algorithmen und Systematik fehlen, ergänzt FlexiConc diese. Auf dessen Basis können einzelne Queries mithilfe ca. 13 verschiedener Algorithmen näher untersucht werden. Zu Beginn der Analyse kann also ein bestimmter Algorithmus ausgewählt werden und anschließend mit anderen Algorithmen und deren Ergebnisse verglichen werden. Dabei steht im Vordergrund, dass CADS möglichst objektiv durchgeführt werden kann und keine möglichen Interpretationen der Ergebnisse durch die Wahl der Algorithmen verhindert werden. Nach dem Prinzip Selecting, Ordering (Sorting und Ranking) und Grouping (Partinioning und Clustering) wird in FlexiConc zunächst mit einer Query nach Konkordanzen und ihren zugehörigen Mustern „colligation, semantic preference and semantic prosodie“ (Dykes et al., 2025b) gesucht. Dabei hilft die KWIC (key word in context) Ansicht erheblich. In weiteren Schritten wird dann mithilfe der eben genannten Prinzipien nach Kollokationen gesucht. Somit wird manuelles Heraussuchen solcher Kollokationen durch das Tool erspart. Insgesamt bietet FlexiConc damit viele Analysemöglichkeiten für Konkordanzen mitsamt mathematischem Rahmen für verschiedene Algorithmen. In Kombination mit einem Konkordanzer der Wahl können somit qualitativ hochwertige Analysen durchgeführt werden. Außerdem bietet FlexiConc neben einer Vielzahl verschiedener Algorithmen außerdem einen Analysistree, durch den die vollzogene Analyse dokumentiert werden kann (Dykes et al., 2025a). Somit wird die Analyse reproduzierbar und transparent. Im Kapitel 3.5 wird die Funktionsweise von FlexiConc genauer beschrieben.

3. Sammlung und Vorverarbeitung der Daten

Die Daten von Donald Trump und Elon Musk sind aufgrund ihrer politischen Entwicklung besonders interessant. Beide begannen ihre Social Media Karriere als Unternehmer. Trump startet 2011 seine politische Karriere und erhöht Schritt für Schritt sein Pensum an Posts. Im Schnitt postet Trump 16 Posts am Tag; gemessen am Postingverhalten gewöhnlicher Bürger oder anderer Politiker ist das sehr viel. Bei Elon Musk ist die Anzahl an Posts erst mit dem Kauf von Twitter 2022 drastisch angestiegen. Danach verfolgt Musk 2024 vorrübergehend eine politische Laufbahn an der Seite von Trump ein. Er unterstützt Trump

bei der US Wahl 2024, bei welcher dieser zum zweiten Mal Präsident wird. Nach mehreren Vorfällen wendet sich Musk im Juni 2025 allerdings gegen Trump. Aufgrund dieser politischen Schwankungen ist es meiner Meinung nach relevant, die Social Media Daten der beiden Persönlichkeiten zu analysieren und deren Kommunikationsstrategien genauer zu betrachten. Deshalb erfolgt im nächsten Abschnitt eine Beschreibung der Datenerhebung, die einen notwendigen vorbereitenden Schritt für die Analyse darstellt.

3.1 Beschaffung der Daten

Für eine sinnvolle Analyse der Socialmedia-Kommunikationsstrategien von Donald Trump und Elon Musk sind ausreichend qualitativ hochwertige Daten nötig. Duplikate, unvollständige oder fehlende Posts können das Bild, das durch die Analyse entsteht, verzerren. Für Elon Musks Tweets gab es glücklicherweise bereits einen geeigneten Datensatz. Auf Kaggle werden von Dada Lindell zwei CSV-Files (*all_musk_posts.csv* - Elon Musk's tweets from his official account (@elonmusk) from the very beginning till April 13, 2025, *musk_quote_tweets.csv* - the original tweets that Elon Musk quote-tweeted to his official account (@elonmusk) from the very beginning till April 13, 2025) zur Verfügung gestellt (Lyndell, 2025). Die darin enthaltenen Daten bestehen aus mehreren verschiedenen Datensätzen (All Elon Musk's Tweets, tweets from Bill Gates, Elon Musk and Ed Lee, Elon Musk Tweets, 2010 to 2017, Elon Musk Tweets (2021-2023)), die ebenfalls auf Kaggle zugänglich sind. Zudem wurden die neuesten Daten bis April 2025 zusätzlich durch den Autor gescrapt und hinzugefügt. Leider sind die Daten aufgrund der Kombination verschiedener Korpora nicht besonders einheitlich strukturiert. Bei einem Teil der Daten (vor allem die 802 neuesten Tweets) fehlen Angaben zu der Anzahl an Likes, Retweets, Replies und vieles mehr. Vermutlich wurde zwischenzeitlich etwas an der Twitter (X) API geändert, sodass nicht mehr alle Posts beim automatischen Scraping richtig erfasst wurden. Dies könnte dem Autor der beiden Korpora entgangen sein, da entsprechende Hinweise in der Beschreibung der Korpora fehlen. Mangels eines gleichwertig umfangreichen Korpus wird dieses, trotz seiner leichten Unvollständigkeit, zur Analyse herangezogen. Die jeweiligen Texte der einzelnen Tweets sind überwiegend vollständig, wobei 845 Posts abgeschnitten und nicht korrekt im csv File gespeichert wurden. Da 845 Posts nur etwa 1,55 Prozent der gesamten Daten darstellen, ist diese Fehlerrate für eine sinnvolle Korpusanalyse noch akzeptabel, weshalb die betroffenen Posts trotzdem im Korpus stehen lasse. Außerdem handelt es sich bei *all_musk_posts.csv* um einen umfangreichen Datensatz mit 54.461 Tweets über den Zeitraum von 2010-2025 (April) hinweg, sodass es relativ gut mit dem Datensatz von Trumps Social Media vergleichbar ist. Dieser erfasst den Zeitraum von 2011 bis 2025 (August) und enthält circa 88.000 Posts. Die Anzahl der Posts beim Trump-Datensatz ist zwar fast doppelt so groß, dafür umfassen die beiden Datensätze von Trump und Musk allerdings einen ähnlichen Zeitraum. Somit unterliegen sie denselben politischen und gesellschaftlichen Veränderungen und referieren auf dieselben Ereignisse, wodurch sie besser verglichen werden können. Was den Datensatz zu Trumps Social Media Posts angeht, so war die Sammlung der Daten um einiges aufwändiger als die von Musks Posts, da kein vollständiges Korpus im Vorfeld zur Verfügung gestellt

wurde. Die Webseite des *Trump Twitter Archive (kurz: TTA, Brown & Williams, 2024)* stellt die Daten der Trump Posts von 2011 bis 2021 in Form eines json- oder csv- Files zum Download zur Verfügung. Diese Daten wurden von den Autoren des Datensatzes mithilfe eines automatischen Scraper für Twitter gescrapt (Brown & Williams, 2014). Der Scraper wurde mit Selenium und Tweepy konzipiert. Selenium hilft dabei, den Browser zu öffnen und durchgehend zu prüfen, ob neue Posts von Trump hochgeladen wurden. Ab 2021 wurde allerdings die API von Twitter verändert, sodass man nun für den Zugang zur API teuer bezahlen muss, während die API vorher kostenlos war. Seit Trumps Wahlniederlage 2021 und dem historischen Sturm aufs Kapitol wurde Trump, aufgrund eines „anheizenden Kommentars" zunächst auf Twitter gesperrt (Schmidt, 2022). Später wechselte er auf seine eigens gegründete Plattform Truth Social, welche über keine API verfügt (Brown & Williams, 2024). Dies erschwerte das Scraping der Daten, sodass die Autoren des TTA händisch alle weiteren Posts von 2021 bis zum 4. November 2024 extrahierten. Dieser Teil der Daten wird leider nicht in Form eines csv oder json-files zur Verfügung gestellt, sondern muss einzeln und kleinschrittig (immer 2.000 Posts auf einmal) in json-format umgewandelt und dann heruntergeladen werden. Die Umwandlung in json-format wird durch die Webseite angeboten, allerdings dauert es lange, alle Daten von 2021 bis 2024 Stück für Stück umwandeln zu lassen. Außerdem ist das json-Format nicht ganz korrekt formatiert, was eine weitere Verarbeitung erforderlich macht. Die Daten ab November 2024 fehlen auf der Webseite vollständig und wurden bislang nicht zur Verfügung gestellt – vermutlich, weil das Extrahieren der Daten erneut erschwert wurde. Jedoch stellt die Webseite *Rollcall* (Fiscal Note Inc., 2025) alle 88.000 Posts von 2011 bis 2025 tagesaktuell zur Verfügung. Auf der Webseite ist zudem eine Suchmaske implementiert, mit der die Posts analysiert werden können. Da die Daten für die Bachelorarbeit in einem Korpus gespeichert werden müssen, müssen sie zu diesen Zwecken heruntergeladen werden. Dafür schrieb ich einen Scraper mit den Bibliotheken Playwright und aiohttp, der die fehlenden Posts (von November 2024 bis August 2025) herunterlädt und als csv file speichert. Die Variabilität der verschiedenen APIs erfordert eine kontinuierliche Anpassung des Scrapers (Gul et al., 2025). Für die bereits herunter geladenen Daten ist das jedoch nicht mehr nötig. Es werden dementsprechend nur die Daten bis zum 4. November 2024 heruntergeladen. Ab 7.000 Posts werden immer mehr Duplikate extrahiert, was die Daten verfälscht. Redundanz ist speziell im Netz ein Problem: Trotz großen des Aufwands, sie zu vermeiden, entstehen dennoch Duplikate oder Beinahe-Duplikate (Boleda et al., 2006). Die Webseiten sind oft nicht einheitlich strukturiert oder weisen an manchen Stellen unvollständige Daten auf (Gul et al., 2025), sodass Probleme wie Duplikate oder andere Fehler häufig vorkommen können. Daher ist eine entsprechende Nachbearbeitung der Daten ausschlaggebend, um die Qualität der Daten sicher zu stellen. Wie genau vorgegangen wurde, wird im folgenden Kapitel erklärt.

3.2 Scraping mit Playwright

Nun wird die Funktionsweise meines Scrapers beschrieben und wie die Ergänzung der fehlenden Daten von November 2024 bis August 2025 gelang.

Bevor mit dem Scraping begonnen werden kann, sollte sich immer mit der Webseite (in dem Fall Roll Call, Fiscal Note Inc., 2025) und deren HTML-Struktur vertraut gemacht werden. Den Quellcode kann man sich mit einen Linksklick der Maus und dem Button auf „Quellcode“ anzeigen lassen. Hier ist farbig hinterlegt, welche Teile des Codes welche Informationen beinhalten. Bei der Inspektion des Quellcodes stellt sich heraus, dass alle Informationen in einzelnen Blöcken „Blocks“ gespeichert sind. Auf der linken Seite der Webseite finden sich Screenshots der Originalposts von Truth Social und auf der rechten Seite sind Informationen zur Plattform, dem Autor, Datum, Uhrzeit und Text hinterlegt.

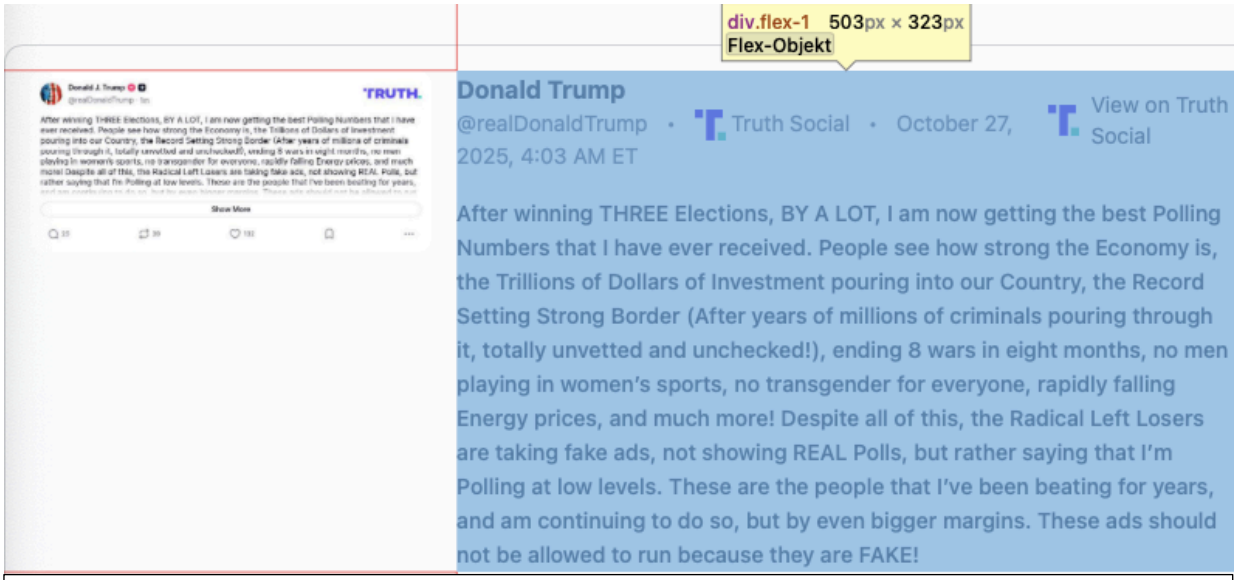

Abbildung 1: Blick auf die Webseite Roll Call Factbase mit in blau unterlegtem Textblock

Abbildung 2: HTML-Struktur der Webseite

```
<div class="flex flex-col md:flex-row gap-4"> flex
  <template x-if="item.image_url">…</template>
  <div class="w-full md:w-1/3">…</div>
  <div class="flex-1">
    <div class="flex flex-col md:flex-row items-start
    md:items-center justify-between mb-4 gap-4">…</div>
    flex
    <template x-if="item.social.post_html">…
    </template>
    <div class="text-sm font-medium whitespace-pre-
    wrap leading-relaxed text-[#2F3C4B] [&>iframe]:!w-
    full [&>iframe]:!h-auto" x-html=
    "item.social.post_html">…</div>
    <template x-if="!item.social.post_html">…
    </template>
    </div>
  </div>
 </div>
</div>
```

Da sie alle in einem gemeinsamen Block gespeichert sind, müssen die Informationen mithilfe von RegEx gesucht, gefunden und in verschiedenen Listen gespeichert werden. Da die Seite außerdem nicht statisch ist, sondern (vermutlich auch aufgrund der großen Menge an Posts) mit JavaScript nachlädt, muss ein Scraping Tool gewählt werden, das damit umgehen kann. Wie im Kapitel *2.1 Web Scraping* ausführlich erläutert, entscheide ich mich für Playwright, ein moderneres, eher unbekanntes Tool, welches allerdings einige praktische Methoden implementiert hat und fast doppelt so schnell wie Selenium läuft. Um zu wissen, wie die Daten genau extrahiert werden sollen, ist es erforderlich, schon vor dem Scraping zu überlegen, wie die Daten am Ende aussehen sollen. Ich entscheide mich dafür, alle erreichbaren Daten (Autor, Plattform, Datum, Uhrzeit, Text – ohne die anderen Informationen und Bild – also den Screenshot) zu extrahieren. Zur besseren Korpusanalyse später wird das zudem Datum in Tag, Monat und Jahr aufgeteilt und durch eine ID ergänzt, um die Posts durchzuzählen.

Um zunächst sicher zu stellen, dass der Text richtig getrennt wird, wird die Methode *extract_metadata_text* definiert, die den Text in die verschiedenen Informationen teilen soll, nachdem mit einer RegEx nach dem Textblock gesucht wird. Wie im Quellcode zu sehen ist, sind die Informationen zu Autor, Plattform, Datum und Uhrzeit in der ersten Zeile gespeichert. In der zweiten Zeile folgt der Text „View on Truth Social“. Um hier möglichst viel Flexibilität zu gewährleisten, wird in einer Schleife geprüft, ob die Zeilen mit „View“ beginnen. Wenn also kein weiterer Text folgt, oder dieser leicht variiert, sollte der Text trotzdem richtig erkannt werden. Ab Zeile 3 beginnt der Text, der auch in einzelne Absätze unterteilt sein kann. Bei einem der ersten Versuche wurden nur die jeweils ersten Absätze erkannt und weitere Absätze wurden als neuer Post gespeichert, was ziemlich viele Duplikate erzeugte. Mit einer kleinen Testfunktion wird ein Beispielpost getestet, in welchem alle möglichen Schwierigkeiten, die beim Scraping auftreten können (wie zum Beispiel Absätze, usw.), berücksichtigt werden. Wenn die Kategorien Autor, Plattform, Datum, Uhrzeit und Text richtig erkannt werden, ist die Methode mit hoher Wahrscheinlichkeit keine Quelle weiterer Fehler. Da sehr viele Daten heruntergeladen werden müssen, arbeite ich im async-Modus, welcher durch Playwright zur Verfügung gestellt wird. Somit können die laufenden Prozesse asynchron bearbeitet werden, ohne auf eine feste Reihenfolge achten zu müssen. Das beschleunigt den Prozess.

Was passiert nun genau im Code? Zunächst wird mit SSL getestet, ob der verwendete Server vertrauenswürdig ist (Zertifikatsprüfung und Hostname). Außerdem wird ein Ordner „Images“ erstellt, in dem die Screenshots gespeichert werden können. Wenn das Programm abbricht und neu gestartet wird, soll dieser Ordner nicht überschrieben werden, sondern um neue Bilder ergänzt werden. Die spätere Datei, in der die extrahierten Posts gespeichert werden sollen, wird ebenfalls zu Beginn des Codes erstellt: *factbase_posts_clean.csv*. Ich entscheide mich für einen csv-File. Allerdings kann das Format auch schnell in *json* umgewandelt werden. Die restlichen Daten zur Trumps Posts aus dem Trump Twitter Archiv liegen nämlich im json-Format vor.

Die Methode *download_worker* ermöglicht das parallele Herunterladen der Daten, da das Programm durch die Extraktion der Screenshots sehr langsam wird und etwa drei Stunden für 500 Posts brauchte. Mit der neuen Methode können diese Prozesse beschleunigt werden. Die Screenshots werden im Ordner *Images* gespeichert und sobald die erstellte Session gültig ist (resp.status==200), werden neue Bilder heruntergeladen. Die Variable für die Anzahl an Workers kann beliebig angepasst werden, in meinem Fall sind es 10. Wenn das Herunterladen der Screenshots nicht funktioniert, wird eine Fehlermeldung angezeigt.

Um Duplikate so gut wie möglich zu vermeiden, wenn das Programm scrollt und sowohl bereits gespeicherte als auch neue Posts auf der Seite angezeigt werden, wird die Methode *make_post_key* definiert. Diese Methode soll einen eindeutigen Schlüssel für jeden Posts generieren, der dann in der *scrape_all_dynamic* Methode verwendet werden kann, um bereits gesehene von neuen Posts zu unterscheiden. Für den Schlüssel werden zunächst Kategorien ausgewählt, die sich im Laufe der Posts unterscheiden. Der Autor sollte nämlich – sofern keine Fehler enthalten sind – immer identisch sein und eignet sich nicht für einen Schlüssel. Der Einfachheit halber wurde „Plattform“ zum Key hinzugefügt, obwohl auch hier meistens „Truth Social“ stehen sollte. In diesem Fall macht es jedoch keinen Unterschied in der Laufzeitdauer und bietet ein weiteres Kriterium, anhand dessen der Key verglichen werden kann. Die beiden wichtigsten Kategorien sind „date“ und „time“, die in der Kombination miteinander einzigartig sein sollten. Da auf der Webseite „Rollcall“ allerdings auch von Trump gelöschte Posts aufzufinden sind, kann es durchaus sein, dass ein gelöschter und ein echter Post am gleichen Tag zur selben Uhrzeit veröffentlicht wurden. Da Roll Call (Fiscal Note Inc., 2025) keine Millisekunden speichert, sind die Posts dadurch nicht unterscheidbar. Abgesehen von den Grunddaten Plattform, Datum und Zeit gibt es zudem zwei optionale Kategorien „text“ und „img“, die nur Teil des Keys werden, wenn man sich bei den Parametern *include_image* und/ oder *include_text* dafür entscheidet. Bilder einzubeziehen, sorgt dafür, dass der Key sicherer wird bzw. der Key einzigartiger wird; allerdings lädt das Programm dann sehr langsam, sodass ich Bilder in der finalen Fassung nicht einbeziehe. Auch im Text wiederholt sich Trump des Öfteren (z.B. „Make America great again“, insgesamt 16 Mal). Trotzdem wird der Text in der finalen Fassung einbezogen, da der Text einen guten Indikator für Duplikate bildet und zusammen mit Datum, Uhrzeit und Plattform einen einzigartigen Key darstellt. Alle vier (optional fünf) Kategorien werden mit Trennstrichen aneinandergehängt und mit *md5* verschlüsselt.

In der letzten Methode *scrape_all_dynamic* werden alle Prozesse zusammengeführt und der Scraping Prozess wird durchgeführt. Zunächst wird eine leere Liste für die zukünftigen Posts erstellt und *seen_posts = set()* gesetzt. Im nächsten Schritt wird geprüft, ob die Datei *factbase_posts_clean.csv* bereits existiert - wenn das Programm beispielsweise mehrere Anläufe braucht, um alle Posts zu extrahieren -, da der ganze Fortschritt nicht gelöscht bzw. überschrieben werden soll. Playwright wird im asynchronen Modus gestartet und eine HTTP-Session über *aiohttp* erstellt. Über *aiohttp* kann ähnlich wie bei *Requests* eine Session eröffnet werden, in der das HTTP-Protokoll abgefragt wird, um die öffentliche API der Seite zu

bekommen. Ein Scraper muss immer mit einer solchen Bibliothek kombiniert werden, um von der Webseite Informationen zu erhalten. Als Browser wird Chromium gewählt (einer von drei möglichen Browsern, mit denen Playwright zusammenarbeitet). Der Browser wird *headless* (also ohne sichtbares Fenster) gestartet. Im nächsten Schritt wird eine neue Seite aufgerufen. Im nächsten Schritt wird die Webseite von Rollcall aufgerufen. *Sleep (2)* ist wichtig, damit der Scraper möglichst menschenähnlich wirkt und nicht zu viele Anfragen auf einmal stellt. Im Anschluss daran sollen alle div.block, die sich auf der Seite befinden, ausgewählt werden. Wenn das nicht funktioniert und keine neuen Blöcke gefunden werden, bricht das Programm nach einer Meldung (über print) ab. Sofern neue Blöcke gefunden werden, werden die Posts gezählt und mit dem gesetzten Maximum von 90.000 Posts abgeglichen. Wenn das Maximum erfüllt ist, bricht das Programm ab. Das Maximum wurde in den Code implementiert, um eine Abbruchbedingung zu schaffen. Da der Code allerdings nur für die ersten 7.000 Posts funktioniert und danach nur langsam vorankommt und sehr viele Duplikate (trotz des Keys) speichert, wurde die Abbruchbedingung nie ausgetestet. Der Code brach immer vorher ab, wenn keine neuen Blöcke auf fünf aufeinander folgenden Seiten gefunden wurden. Im nächsten Schritt wird jeder neue Post gespeichert (in einzelnen Listen für jede Kategorie). Die einzelnen Bilder werden im Hintergrund über *Queue asynchron* heruntergeladen und gespeichert, sodass das Programm effizienter arbeiten kann (und nicht zu lange auf das Herunterladen der Bilder warten muss). Für den Post wird nun der Key über die Methode *make_post_key* erstellt und mit den bereits gesehenen Daten (*seen_posts*) abgeglichen. Wenn er noch nicht gesichtet wurde, wird er zu *seen_posts* hinzugefügt und der Post wird in *data* gespeichert. Im Anschluss daran scrollt der Scraper und zählt mit *no_new_rounds*, auf wie vielen Seiten keine neuen Posts gefunden wurden. Der Parameter *max_no_new=5,* welcher der Methode *scroll_all_dynamic* mitgegeben wurde schafft eine Abbruchbedingung, wenn auf 5 Seiten keine neuen Posts gefunden wurden. Falls die Seiten langsamer über Java Script nachgeladen werden als *await asynchio.sleep(3)* wartet, bricht das Programm ab, obwohl noch Posts auf der Webseite vorhanden wären. Wenn das der Fall ist, kann der Code problemlos neu gestartet werden, da zu Beginn von *scrape_all_dynamic* getestet wird, ob bereits Daten in der Datei gespeichert wurden. Auch neue Keys, für die bereits in *data* befindlichen Posts werden erstellt, da diese nach dem Abbruch des Scrapers nicht gespeichert werden können. Nach dem Scrollen werden die Prozesse im *Queue* beendet und der Browser geschlossen. Im Anschluss werden IDs vergeben und die gespeicherten Posts als csv gespeichert.

Sobald die Daten vollständig heruntergeladen wurden – also von November 2024 bis zum August 2025 – können sie mithilfe von Pandas inspiziert werden. Außerdem sollten die einzelnen Einträge stichprobenartig auf Richtigkeit und Vollständigkeit überprüft werden. Duplikate (es sind trotz der Keys 1.984 Duplikate) werden nachträglich entfernt.

Da die Daten zu Donald Trumps Social Media Posts aus mehreren Teilen kombiniert werden müssen und die anderen Dateien als json vorliegen, wird auch die gescrapte Datei in json umgewandelt. Die restlichen Daten des Trump Twitter Archivs werden mit *glob* aneinandergehängt. Leider sind diese einzeln in

2.000er Schritten herunter geladenen Daten nicht richtig als json-Format gespeichert, da auch einige HTML-Elemente mitgespeichert wurden. Daher müssen die Daten erst bereinigt werden und können anschließend zusammen gefügt werden.

3.3 Vorverarbeitung (auch für CWB)

Um die Daten als Korpus über die IMS Korpusworkbench (CWB) indexieren zu können, muss noch eine Vielzahl an weiteren Vorverarbeitungsschritten durchgeführt werden. Alle Schritte sind in den Jupyternotebooks 01 und 03 zu finden, welche im Repository „TrumMus“ auf GitHub hochgeladen wurden.

Nur selten wird über die Schwierigkeit der Vorverarbeitung gesprochen. Doch sie nimmt bei den meisten Projekten einen großen Teil der Bearbeitungszeit in Anspruch, da die verschiedensten Schnittstellen aneinander angepasst werden müssen (Kunilovskaya & Plum, 2021). Speziell Twitter-Texte beinhalten Sparse und Noisy Data (Derczynski et al., 2013). Im Fall der Bachelorarbeit mussten die gescrapten Daten für Trumps Posts von Duplikaten gereinigt werden. Hier überlappten sich auch die Angaben für die jeweiligen IDs, welche deshalb neu nummeriert werden mussten. Außerdem müssen fehlende Werte ergänzt werden: Die Spaltenanzahl der gescrapten Daten und der von TTA herunter geladenen Daten waren nicht identisch, sodass fehlende Werte mit „Nan“ oder „None“ aufgefüllt werden sollten. Die Daten des TTA haben folgende Spalten: ['id', 'text', 'datetime', 'favorites', 'isRetweet', 'retweets', 'isDeleted', 'device', 'isFlagged']. Da die gescrapten Daten die Spalten „date“, „time“, „day“, „month“, „year“ aufweisen, werden die Daten des TTA ebenso aufgetrennt. Die Spalten „author“ und „image_path“ lasse ich weg, da der Autor durchgehend Trump ist und die Bilder nicht für die weitere Analyse der Kommunikationsstrategien verwendet werden. An den Stellen, an denen bisher keine Werte vorhanden waren, werden die fehlenden Werte mit null aufgefüllt. Das Ergebnis wird im json-Format als *tta_final_clean.json* gespeichert.

Außerdem steht zum Beispiel bei machen Posts zu Beginn „RT“, was für „Retweet“ steht. Wenn also „RT“ beim Post steht, sollte in der Spalte „isRetweet“ „True“ erscheinen. Ebenso sind alle Daten bis 2021 auf Twitter gepostet worden, danach wurde Trump auf Twitter gesperrt und erst im November wieder zugelassen. Im Februar 2022 kündigte er Truth Social, seine neue Plattform, an. Seitdem postete er fast ausschließlich auf Truth Social. Da beim Zusammenführen der Daten nur die gescrapten Daten Informationen zur Plattform enthielten, wurden Informationen zur Plattform nachträglich hinzugefügt. Abgesehen von diesen Feinheiten wurden auch HTML-Tags entfernt, die bei den von TTA herunter geladenen Daten noch stellenweise vorhanden waren. In den Daten des TTA, die von Twitter stammen, wurde das Datum als „Snowflake“ gespeichert (mit Millisekunden), sodass der Datentyp an die gescrapten Daten angepasst werden muss. Somit ist der Datensatz zu Trump fast fertig.

Um den Datensatz allerdings in die IMS Korpusworkbench (CWB) einlesen und als Korpus indexieren zu können, müssen noch weitere Schritte erfolgen. CWB akzeptiert verschiedene Datenformate, mit denen das Korpus eingelesen werden kann – unter anderem vrt oder xml. Im Rahmen dieses Projekts

entschied ich mich für vrt – also „verticalized text“. Außerdem müssen xml-empfindliche Zeichen wie &“<> escaped werden – beispielsweise mit & -> &. Beim Einlesen in die Korpusworkbench können die Daten mit der Ergänzung „-x“ im Befehl wieder umgewandelt werden.

3.4 POS-Tagging mit Stanza/Tweebank

Im folgenden Kapitel wird eine kleine Auswahl der eben beschriebenen Tagger getestet – nämlich fünf Tagger mit kleineren Variationen, um zu vergleichen, wie Tagger, die auf normalsprachlichen Texten mit Tweets umgehen im Vergleich zu Taggern, die auf Tweets trainiert wurden oder dafür Modifikationen aufweisen, abschneiden.

Um die aufbereiteten Daten entsprechend linguistisch analysieren und auswerten zu können, sollten die Tweets vorher mit position tags versehen werden, sodass zu den einzelnen Wörtern und Token der Tweets jeweils Wortarten gespeichert werden. Da es sich um Tweets handelt, schneiden herkömmliche Tagger leider nicht unbedingt gut ab, wie bereits im Kapitel *2.2 POS-Tagging* erwähnt. Aufgrund des stark vereinfachten Satzbaus und der Knappheit der Tweets ist es schwer, die Tweets auf dieselbe Art wie lange Romane zu taggen. Neben einer knappen Schreibart sind Tweets zudem eher an der gesprochenen Sprache orientiert, sodass Umgangssprache und Abkürzungen häufiger Verwendung finden. Außerdem gibt es typische Tweet-Konventionen wie etwa #Hashtags oder Erwähnungen von Personen mit @ und deren Usernamen, was die Verwendung der echten Namen ersetzt. Vermehrt werden zudem Emojis verwendet, für die es bei herkömmlichen Taggern kein Tag gibt, da sie in Romanen oder Zeitungsartikeln nicht vorkommen. Aufgrund der verschiedenen Arten, wie die Tagger mit tweet-spezifischen Tags umgehen und weil viele nicht speziell auf Tweets trainiert wurden, ist unklar, welcher Tagger am besten performt. Auf normalsprachlichen Texten performen alle aufgeführten und später getesteten Tagger sehr gut – zwischen 95 und 97%. Allerdings gibt es bisher kaum einheitliche Zahlen zur Performance auf Tweets. Aufgrund der unübersichtlichen Menge an Tokenizern, Taggern und Parsern und deren ständiger Updates oder Verbesserungen ist es schwer zu überblicken, welche Modelle auf welche Weise kombiniert werden müssen, um die besten Ergebnisse zu erzielen. Aufgrund dieser Problematik wurden fünf verschiedene Tagger mit vortrainierten Modellen, die folglich Tokenizer, Tagger und Parser bereits beinhalten, mithilfe eines kleinen Testdatensatzes von 50 Tweets getestet und verglichen.

Das Testkorpus beinhaltet verschiedene Schwierigkeiten, mit denen ein Tagger bestenfalls umgehen können sollte. Um diese verschiedenen Schwierigkeiten alle zu erfassen, wurden sieben Funktionen geschrieben, die Tweets finden sollen, die Erwähnungen (@), Hashtags (#), URLs (https://...), Emojis (in Unicodezeichen) und eventuell häufige englische Rechtschreibfehler enthalten. Außerdem wurde auch nach Tweets mit einer Länge von über 200 Zeichen gesucht, um zu sehen, wie die Tagger mit längeren Tweets umgehen können. Das ist vor allem bei den Taggern relevant, die speziell auf kurze Tweets trainiert wurden. Von jedem Kriterium wurden 5 zufällige Tweets ausgewählt. Der Rest der Tweets wurde für das Testkorpus bestehend aus 50 Tweets einfach zufällig aufgefüllt. Zur Erstellung des Testkorpus wurden

die Tweets von Trump verwendet. Mit dem Parameter *„random_size*“, dem jeder der sieben Funktionen mitgegeben wurde, wird die Reproduzierbarkeit gewährleistet. Wenn der Code erneut ausgeführt wird oder von anderen Personen nachvollzogen werden soll, bietet es sich an, den Code reproduzierbar zu gestalten, was mit „*random_size*“ gut funktioniert. Getestet werden nun anhand des Testkorpus fünf verschiedene Tagger mit jeweils möglichen Varianten oder Kombinationen. Die Tagger wurden bereits im dazugehörigen Kapitel „POS-Tagging“ des Forschungsüberblicks beleuchtet, weshalb ich in diesem Kapitel nicht näher auf ihre Funktionsweise eingehen werde. Ich werde allerdings auswerten, wie die jeweiligen Tagger in Bezug auf das Testkorpus abschnitten. Um eine nachvollziehbare Auswertung zu ermöglichen, wurden natürlich alle Varianten der Tagger auf demselben Testkorpus getestet.

Die Texte müssen zunächst segmentiert/tokenisiert, ggf. lemmatisiert und schließlich getaggt werden. Nicht alle Tagger performen bei allen drei Aufgaben gleich gut, sodass im Folgenden auch versucht wird, diese einzelnen Arbeitsschritte getrennt voneinander zu analysieren. Die Entscheidung, welcher Tagger gewählt wird, wird dann aber in Bezug auf Praktikabilität und die Zusammenarbeit dieser drei Aufgaben getroffen. Außerdem werden zwei verschiedene Tagsets (UD und Penn Treebank) und eine modifizierte Variante (xpos) verwendet. Im Folgenden werde ich fünf verschiedene Tagger mit teilweise kleinen Modifikationen vorstellen und diese im Hinblick auf verschiedene Wörter (kursiv geschrieben) auswerten. Ausführlicher wird dies im Notebook 02 beschrieben. Hier in der Bachelorarbeit beschränke ich mich jedoch auf ein paar Faktoren, die meinen persönlichen Eindruck wiedergeben.

SpaCy ist das erste getestete Modell. Es liefert alle Funktionen (Tokenisieren, Lemmatisieren, Taggen) mit, wurde aber nicht speziell auf Tweets trainiert, sondern auf Zeitungsberichten und längeren (Web-)Texten. Das verwendete Modell ist „en_core_web_sm“ und das Tagset basiert auf Universal Dependencies. Das Tokenisieren funktionierte sehr gut – allerdings wurden gerade Tweet-spezifische Token wie Hashtags und Emojis falsch behandelt. Hashtags wurden falsch getrennt und dann # als SYM und das folgende Wort als PROPN oder NOUN getaggt (Penn Treebank). Hashtags als Eigennamen zu kategorisieren, ist akzeptabel; allerdings wäre es gut, wenn das # nicht vom darauffolgenden Wort getrennt wird. Auch Emojis werden aufgetrennt und in den ganzen Satz eingebettet, sodass die Tags zwischen X, NOUN und PROPN variierten. Es wäre gut, wenn Emojis nicht als Eigennamen behandelt werden würden – das würde die Ergebnisse der späteren Korpusanalyse verfälschen. Erwähnungen mit vorausgehendem @ werden gut erkannt und meist im Ganzen gelassen. Getaggt werden sie entweder als X (unbekannt), als NOUN oder PROPN. NOUN oder PROPN wären akzeptabel und nicht bedeutungsverzerrend. Im Folgenenden werden nun einige Wortbeispiele herausgegriffen und kursiv geschrieben, um sie im Textfluss erkenntlich zu machen. *Barr's* wird zu *Barr* als PROPN und *'s* als PART tokenisiert und getaggt. Die Lemmatisierung war nur teilweise korrekt, da einige Wörter nicht auf ihren Ursprung zurückgeführt werden konnten: *Illnesses* bleibt auch als Lemma *Illnesses*, aber *Shreds* wird zu *shred* und als VERB erkannt. *L.L. Bean* wird als Token gelassen und nicht weiter aufgetrennt. Auch bei Rechtschreibfehlern performt SpaCy gut: „*former.Miss Alabama"* wird zu: „*former* ADJ. PUNCT *Miss* PROPN *Alabama* PROPN“.

Jedoch werden nicht alle Token richtig getaggt – wie etwa „Meet PROPN The PROPN Press PROPN“ oder auch der Name *Todd*, der als ADJ getaggt wird. *BEST* wird als *BEST* lemmatisiert statt der eigentlichen Grundstufe *good* und als PROPN getaggt. *ol‘* wird als ADJ erkannt. Insgesamt wird an vielen Stellen gut lemmatisiert*: elected* wird zu *elect*, *am* zu *be*, *MADE* zu *make*, *Stores* zu *store*. Beim zweiten Anlauf mit SpaCy wurde der Code ein bisschen angepasst. Hashtags wurden wie bei Versuch Nr. 1 getrennt und in den ganzen Satz des Tweets eingebettet, sodass verschiedene Tags vergeben wurden – meistens aber PROPN oder NOUN, was akzeptable Tags für Hashtags sind, wenn es keine tweet-spezifischen Tags gibt. Erwähnungen werden auch dieses Mal in den Satz eingebettet und als PROPN, NOUN, X, ADJ oder VERB getaggt. Dadurch würden Ergebnisse der Korpusanalyse stark verfälscht werden. Erwähnungen bleiben vollständig. Das Lemmatisieren funktioniert im Allgemeinen besser als beim ersten Versuch: *her* wird zu *she*, *being* zu *be* und *MADE* zu *make*. Allerdings passieren trotzdem Fehler wie BEST, was zu BEST (statt good) lemmatisiert wird. Emojis werden ebenfalls als Teil des Satzes analysiert und erhalten die Tags X, NOUN oder PROPN. Allerdings werden sie gut tokenisiert. In einem weiteren Versuch wurde der Programmcode um eine Twitterfunktion ergänzt, die Hashtags und Erwähnungen erkennen soll. Hier bleiben die Hashtags - im Gegensatz zu den beiden vorherigen Versuchen - vollständig *(#AGENDA47* statt *# AGENDA47*) und werden mit NOUN oder PROPN getaggt. Allerdings werden manche URL zerteilt und das Tokenisieren bzw. Abtrennen der Wörter von darauffolgenden Satzzeichen funktioniert deutlich schlechter als zuvor. Das Tokenisieren funktioniert schlechter, weil die Art und Weise, wie die Funktion eingebaut wurde, dafür sorgt, dass nicht mehr alle Regeln, die SpaCy gelernt hat, angewendet werden, da sie von der Funktion außer Kraft gesetzt werden. Beim vierten Versuch wurden Funktionen für tweet-spezifische Dinge wie Hashtags, Erwähnungen, Emojis und URLs geschrieben. Hier bleibt die Funktionsweise jedoch erhalten und werden lediglich durch die vier Funktionen ergänzt. Es werden zudem spezielle Tags vergeben: für URLs URL, für Emojis NFP, für Hashtags ADD und für Erwähnungen soll PROPN verwendet werden. Hashtags werden erneut getrennt, da sie als Objekt im Satz analysiert werden. Dafür werden alle Erwähnungen, die Teil des Testkorpus sind, richtig tokenisiert und getaggt. Auch Satzzeichen wurden richtig abgetrennt. Emojis wurden meist korrekt tokenisiert und getaggt (als NFP, manchmal aber auch als PROPN). Allerdings funktioniert das Lemmatisieren – wie auch bei den anderen Varianten von SpaCy nur manchmal: *Illnesses* wird zu *Illnesses* lemmatisiert, *her* bleibt *her*, *more* bleibt *more*, während *stolen* richtig zu *steal* lemmatisiert wird. Auch das Tagging funktioniert an einigen Stellen nicht ganz wie gewollt: *JOBS* wird als PROPN getaggt, *Thank you* wird VERB PRON und der Eigenname *Todd* wird als ADJ getaggt.

Flair ist das zweite getestete Modell. Dieses liefert vier Modelle für das Englische, von denen zwei Versionen mit pos (basierend auf Penn Treebank) und zwei mit upos (basierend auf UD) taggt. Zwei Modelle sind english-fast-Versionen und zwei Modelle die Normalfassung. Auf normalsprachlichen Texten funktioniert Flair sehr gut (etwas mehr als 98% F1-Score). Allerdings wurde es nicht speziell auf Tweets trainiert und kann daher mit tweet-spezifischen Tags oder stark vereinfachten Satzstrukturen nicht gut

umgehen. Die größte Schwierigkeit ist vermutlich, dass URL keinen Teil des Satzes ausmachen und Emojis, Erwähnungen und Hashtags nur manchmal Teil des Satzes sind. Das Tokenisieren funktioniert vor allem bei Links gar nicht, sodass diese sehr klein segmentiert werden. Die Lemmatisierung wird bei diesen Modellen nicht mitgeliefert, sodass hierfür im zweiten Anlauf SpaCy verwendet wurde. Auch Hashtags und Erwähnungen werden konsequent aufgetrennt – dafür werden die darin enthaltenen Wörter richtig tokenisiert. Das Tagging funktioniert akzeptabel, trifft aber auch oft nicht zu (vor allem bei Namen werden oft die Tags NUM oder SYM vergeben). Emojis hingegen werden gut tokenisiert und als SYM getaggt, was eine gute Option ist, wenn man sich der UD oder der Penn Treebank bedient und keine speziellen Tags für Emojis implementiert. Flair hat theoretisch auch die Fähigkeit, Mehrwortausdrücke zu erkennen (wie etwa *LL. Bean*). Das funktioniert allerdings nicht konsequent. URLs werden auch aufgrund ihrer starken Segmentierung als eigene Sätze getaggt, was falsch ist und zu viele Tags generiert. Was speziell Tweets angeht, kann Flair also leider nicht mithalten, auch wenn es auf normalsprachlichen Texte gut performt.

Im Allgemeinen wird das Tag NUM zu häufig vergeben und würde so die spätere Korpusanalyse verfälschen: *weekend, Vegas*, *former.Miss* (ein Zeichenfehler), *Illnesses*, *Attorney*, *Shreds* und *Fame* werden als NUM getaggt. Auch der Satz *@MagaGlam<Emojis> Bring Back Trump <Emojis>* wird falsch getaggt: @ X *MagaGlam* X *<Emojis>* SYM *Bring* VERB *Back* SYM *Trump* NUM *<Emojis>* SYM. *This* NUM *week* NOUN *we* VERB *hosted* VERB *a* DET # SYM *MadeInAmerica* NUM *event* NOUN. Das zweite getestete Modell pos/English verhält sich in Bezug auf die Tokenisierung identisch zu upos/english. Obwohl der auf Hugging Face angegebene F1-Score niedriger als der von upos/english ist (98,18 zu 98,6), schneidet das Tagging deutlich besser ab: *This* DT *week* NN *we* PRP *hosted* VBD a DE # NN *MadeInAmerica* NNP *event* NN. Hashtags und Erwähnungen (@) werden weiterhin zerteilt. Emojis werden nicht einzeln tokenisiert, sondern immer paarweise als ein Token gespeichert; aber konsequent mit NFP getaggt. Die dritte getestete Version kombiniert Flair mit SpaCy, da Flair keine Lemmatisierung mitliefert. Flair und SpaCy tokenisieren jedoch verschieden, sodass die jeweiligen Token aufeinander abgestimmt werden müssen. Wenn SpaCy an manchen Stellen nicht mit Flair übereinstimmt, wird zunächst die Funktion .lower() verwendet anstatt des lemmas. Das Lemmatisieren erfolgt daher nur manchmal richtig: *Stores* wird *store*, *being* bleibt *being*, *elected* bleibt *elected*, *workers* wird *worker*, *is* wird *be*, *was* wird *be*, *stolen* zu *steal*, *harder* zu *hard* und *takes* zu *take*. Somit funktioniert die Kombination aus SpaCy und Flair besser als Flair allein. Doch durch das Zerteilen der URLs und dem daraus resultierenden falschen Tagging, kommt Flair als Tagger für die beiden Datensätze von Trump und Musk nicht in Frage.

Als nächsten Tagger stelle ich **TweebankNLP** vor. Version 1.0 implementierte ursprünglich kein POS-Tagging, was sich mit Version 2.0 jedoch änderte. Zur Auswahl stehen zwei für POS-Tagging trainierte Modelle. Die anderen beiden Modelle wurden für NER (namend entity recognition) konzipiert. Ich testete hier nur das Modell, das die beiden Datensätze Tweebank, welches speziell auf Tweets trainiert wurde und EWT (englisches Modell, das Universal Dependencies zur Verfügung stellt) kombiniert. Hier gelingt

das Tokenisieren und Segmentieren nicht besonders gut: bei jedem Wort, das von seinem Hinterglied oder auch einem Satzzeichen getrennt wird, werden zwei @@ hinzugefügt (*@Timc1021* wird zu *@Timc@@ 1021* tokenisiert). Auch *@_KatherineWebb* wird zu *@_KatherineWeb@@* und *b:* tokenisiert; *conceived* wird zu *conc@@ ei@@ ved@@* und auch einige (wenige) URLs werden geteilt. URLs und Erwähnungen werden oft als X getaggt; Emojis und Satzzeichen werden zu <unk> reduziert und als SYM getaggt. An den meisten Stellen gelingt das Tagging gut: *Shreds* wird als VERB getaggt; *Chuck Todd* und *Mariano Rivera* werden als PROPN getaggt und als Mehrworteinheit aufgefasst. Jedoch werden auch andere Ausdrücke, die keine MWEs wären, zusammengelassen. Zudem wird selten richtig lemmatisiert: *Looking* bleibt *Looking, Shreds, being, elected, MADE, skies* und *better* bleiben in ihrer ursprünglichen Form erhalten und werden nicht lemmatisiert.

Die zweite getestete Variante kombiniert TweetNLP mit SpaCy, sodass mit SpaCy lemmatisiert und mit TweetNLP getaggt wird. Das Tokenisieren wird erneut durch TweetNLP bewerkstelligt, was sich in den Ergebnissen widerspiegelt. Die Lemmatisierung gelingt nun jedoch deutlich besser: *Shreds* wird zu *shred* lemmatisiert, *is* wird *be*, *Addressing* wird *address*, *elected* wird *elect*, *being* wird *be*, *MADE* wird zu *make*, *skies* zu *sky* und *better* zu *well*. An den Stellen, an denen falsch tokenisiert oder @@ ergänzt wurde, wurde auch nicht richtig lemmatisiert. Zudem wurden (wie bei der ersten Variante) manchmal URL zerteilt, sodass zu viele Tags vergeben werden. Alle anderen Dinge werden wie bei der ersten Variante behandelt und werden deshalb nicht erneut beschrieben. Die Version mit SpaCy schneidet aufgrund des Lemmatisierens besser ab als TweetNLP allein, jedoch nicht gut genug für das Tagging der beiden Datensätze von Musk und Trump. Der dritte Ansatz ist eine Kombination mit Stanza, sodass Tokenisierung und Lemmatisierung von Stanza übernommen werden. Trotzdem basiert der Code immer noch auf dem vortrainierten Modell von TweetNLP. Nun werden URL ganz gelassen, aber mit X getaggt. Auch das restliche Tokenisieren funktioniert deutlich besser, sodass keine @@ an vorherige Wörter angehängt werden. Das Lemmatisieren funktioniert ebenfalls akzeptabel – überraschenderweise jedoch schlechter als bei der vorherigen Version mit SpaCy: *does* wird zu *do* lemmatisiert und *Comments* zu *comment*, *Looking* zu *look*, *BEST* zu *good, People* zu *person*, *stolen* zu *steal*, aber *Addressing* zu *addressing*, Stores zu *stores* und *Shreds* zu *shreds* (NOUN). Das Tagging ist an einigen Stellen falsch: ol‘ als SYM, on als X, take als NOUN, *jobs* als PROPN, Shreds als NOUN, being als NOUN und for als INTJ. TweetNLP basiert auf Tweebank – kombiniert mit dem Datensatz EWT -, was gegenüber den anderen Taggern, die nicht explizit auf Tweets trainiert wurden, einen Vorteil bietet. Jedoch schneidet es in allen drei Bereichen Tokenisieren, Lemmatisieren und Tagging nicht besonders gut ab. In der Kombination mit SpaCy oder Stanza gelingen Tokenisieren und Lemmatisieren deutlich besser. Der letzte getestete Tagger Stanza bietet die Möglichkeit, eine TweetNLP Pipeline mit tokenize_engine=tokenize/tweet zu ergänzen. Dadurch werden tweetspezifische Daten mithilfe von Tweebank – dem Datensatz von TweetNLP einbezogen.

Kommen wir nun zum letzten getesteten Tagger. **Stanza** ist ein Tagger von der Universität Stanford. Das hier getestete Modell vergab weniger Tags als alle anderen Modelle – insgesamt nur 1190 Tags.

Erwähnungen mit @ wurden meist gut erkannt und so belassen wie sie waren (also nicht aufgetrennt). Emojis werden als PUNCT erkannt, was akzeptabel ist. Es ist sicherlich besser, wenn Emojis als Satzzeichen analysiert werden, als sie im Satz einer variablen Position zuzuweisen. Außerdem erfüllen Emojis in Tweets oder in anderen Social Media Texten (wie WhatsApp) manchmal die Funktion von Satzzeichen. Hashtags werden meist getrennt. URLs werden zwar als Eigennamen getaggt, dafür aber im Ganzen belassen. *Barr's* wird nur zu *Barr* als PROPN, *'s* wird, was akzeptabel ist, da es sich um eine andere Form von *Barr* handelt. *Shreds* wird falsch erkannt und auch der Slang *ol'* wird nicht richtig getaggt und lemmatisiert (nämlich als NOUN, *old*). Was die Lemmatisierung und das sonstige Tagging angeht, schneidet Stanza jedoch besser als SpaCy ab. *Doesn't* etwa wird zu *do* lemmatisiert. Allerdings wird auch hier (ähnlich wie bei *Barr's*) der zweite Teil *not* einfach eliminiert. Kombiniert man Stanza mit Tweebank, sodass der implementierte Tweet/Tokenizer verwendet wird führt zu besseren Ergebnissen. Füttert man diesen also in die Stanza-Pipeline, so sind die Ergebnisse deutlich besser. Das Tokenisieren, Lemmatisieren und Taggen funktionierte sehr gut. Einzelne Abweichungen waren zum Beispiel, dass Hashtags und Erwähnungen dennoch häufig getrennt tokenisiert werden und nur zu etwa 60% der Fälle als ein Token beibehalten werden. Nutzt man das von Stanza speziell für Tweets ergänzte Tagset *xpos*, welches auf UD basiert, aber ADD und NFP als zusätzliche Tags aufweist, so werden die Tweets sehr gut getaggt. URLs werden bei *xpos* als ADD getaggt, bei *pos* als PROPN. *Barr's* wird bei dieser Version aufgetrennt in *Barr* und *'s*. Allerdings werden Emojis meist als Satzzeichen getaggt und nur selten als NFP, was das eigentlich für sie passende Tweet-Tag wäre. Daher wird das Modell im dritten Versuch durch eine Funktion für Emojis ergänzt – der Fehler passiert nämlich konsequent und kann nachträglich korrigiert werden. Die Funktion vergibt bei *xpos* immer dann NFP, wenn ein Emoji erkannt wird. Insgesamt ist diese Version meiner Meinung nach die beste Version, sodass sie für das spätere Tagging der beiden Datensätze Trump und Musk verwendet wird. Auch die Gesamtzahl der vergebenen Tags ist bei der dritten Version bei insgesamt 1207 Tags – ähnlich wie die andere Tagger - wohingegen das reine Stanza Modell nur 1190 Tags vergeben hatte.

Mein persönlicher Eindruck ist also aufgrund der eben genannten Indizien, dass Stanza mit dem Tweet Tokenizer (basierend auf Tweebank) plus Emoji-Ergänzung am besten performt. Obwohl Stanza langsamer als die anderen Tagger läuft, ist die Performance in diesem Fall ausschlaggebend. Dies spricht für die Wahl von Stanza in Kombination mit Tweet Tokenizer. Stanza tokenisiert sehr gut – ebenso wie SpaCy, sodass alle URL vollständig bleiben. Ähnlich wie SpaCy werden Erwähnungen und Hashtags nicht konsistent getaggt, während SpaCy konsistent alle Erwähnungen vollständig beibehält, aber die meisten Hashtags auftrennt. Stanza erfüllt den Mittelweg. Das Tagging funktioniert bei beiden Taggern ähnlich gut. Beide Tagger weisen gelegentlich Fehler auf, vergeben im Großen und Ganzen aber geeignete Tags. Da Stanza allerdings die modifizierte Version der Penn Treebank (xpos) mitliefert, sind hier tweet-spezifische Tags möglich, die SpaCy nicht mitliefert. Beim Lemmatisieren schneidet Stanza besser ab als

SpaCy, obwohl beide auch hier nicht alles korrekt erkennen. Aufgrund der insgesamt besseren Performance werde ich die beiden Datensätze mit Stanza/Tweet-Tokenizer taggen.

3.5 CWB und FlexiConc

Die vollständig getaggten Datensätze müssen im nächsten Schritt als Korpus bei CWB eingelesen werden. Dafür ist das Format vrt (verticalised text) nötig, wie in Kapitel 3.3 beschrieben wurde. Der Schritt, alle Daten in vrt-Format umzuwandeln, geschieht in meinem Code gleichzeitig mit dem POS-Tagging, da die Daten an dieser Stelle sowieso neu formatiert werden müssen. Vorher waren die einzelnen Einträge der Datensätze in Tweets unterteilt. Nach dem Tagging müssen sie in Metainformationen und getaggten Text unterteilt werden, sodass das Auftrennen in <text metadata> und <text> in einem Schritt getan werden kann. Anschließend werden die beiden als *trump.vrt* und *musk.vrt* gespeicherten Datensätze als Korpora bei CWB indexiert. Sobald das Korpus keine Fehlstellen oder kaputten Formatstücke mehr aufweist, kann mit FlexiConc, dem Korpusanalysetool gearbeitet werden. Außerdem wurden beide Korpora nach kleinen Anpassungen auf CQP-Web hochgeladen, sodass auch dort Korpusanfragen bearbeitet werden können. Für die Arbeit mit dem Korpusanalysetool FlexiConc gibt es an verschiedenen Stellen Dokumentationen, mit Hilfe derer die Installation und erste Schritte mit FlexiConc leicht verstanden werden können. Im Notebook *04* sind die einzelnen Befehle, die für das Einlesen und Indexieren der Korpora in CWB nötig waren, erklärt. Auch die weiteren Schritte zur Erstellung von Frequenzanalysen, der Bildung von Subkorpora sowie die Bildung von Kollokationen und Konkordanzen wird in den Notebooks *04-06* beschrieben.

Im Folgenden werde ich die wesentlichsten Schritte zur Analyse mit FlexiConc beschreiben. FlexiConc beruht auf Konkordanzanalysen, für die aktuell 17 verschiedene Algorithmen zur Verfügung stehen – gerade die Art der Berechnung und welcher Algorithmus genutzt wird, verändert und beeinflusst die Ergebnisse der Analyse. Mit *from flexiconc import Concordance* wird eine Klasse Concordance importiert, für die mit *hillary = Concordance()* eine Instanz der Klasse definiert wird. Im weiteren Verlauf werden alle Analysen einer Query in einem Analysistrees gespeichert. Der Analysistree ermöglicht es später, die erfolgte Suche transparent nachvollziehen zu können. Über *display(HTML(generate_concordance_html(hillary, hillary.root, n=1000)))* werden die Konkordanzen der erfolgten Suche tabellarisch in KWIC angezeigt. Mit *display(HTML(generate_analysis_tree_html(hillary)))* wird der Analysistree generiert. Über einen neuen Knoten kann einer der folgenden verschiedenen Algorithmen angewandt werden: *hi1 = hillary.root.add_subset_node(("Random Sample",{'sample_size': 10, 'seed': 9101989}).*

So könnte ein Analysistree beispielsweise aussehen (siehe: Jupyter Notebook 05):

Abbildung 3: Analysistree „hillary“

```
5 display(HTML(generate_analysis_tree_html(hillary)))
```

Query: [lemma="Hillary"]

[0] subset (1088):
[1] subset (20): ⚙ Random Sample {'sample_size': 20, 'seed': 12}
[2] arrangement : ⚙ Partition by Ngrams {'positions': [-1], 'tokens_attribute': 'word', 'case_sensitive': False}
[4] subset (44): ⚙ Manual Line Selection {'groups': ["('beat',)"]}
[3] arrangement : ⚙ Partition by Ngrams {'positions': [1], 'tokens_attribute': 'word', 'case_sensitive': False}
[5] subset (28): ⚙ Manual Line Selection {'groups': ["('is',)"]}
[6] arrangement : ⚙ Partition by Ngrams {'positions': [1, 2], 'tokens_attribute': 'word', 'case_sensitive': False}
[7] subset (0): ⚙ Manual Line Selection {'groups': ["('is',)"]}
[8] subset (36): ⚙ Manual Line Selection {'groups': ["('clinton', 'is')"]}
[9] arrangement : ⚙ Partition by Ngrams {'positions': [1, 2, 3], 'tokens_attribute': 'word', 'case_sensitive': False}

Weitere Schritte der Analyse werden in den Jupyter Notebooks *04-06* aufgezeigt. Außerdem wird die Korpusanalyse nicht nur mit FlexiConc durchgeführt, sondern auch durch cwb-ccc und CQP-Web unterstützt.

4. Korpusanalyse mit CADS

Nachdem das Beschaffen und Verarbeiten der Daten sowie der aktuelle Stand der Forschung zu den jeweils vorgestellten Methoden und Modellen nun ausführlich beschrieben wurde, erfolgt der analytische Teil der Arbeit: die Korpusanalyse. Im Folgenden werden die beiden erstellten Korpora zu den Tweets von Donald Trump und Elon Musk zunächst im Kapitel *4.1 Analyse mit cwb-ccc* mithilfe von cwb-ccc verglichen und anschließend im Hinblick auf verschiedene Kommunikationsstrategien analysiert. Im Kapitel *4.2 Erarbeitung der Kommunikationsstrategien* werden 22 verschiedene Kommunikations- oder auch Manipulationsstrategien vorgestellt. Nach diesen wird dann in den beiden Korpora TRUMP und MUSK genauer gesucht und geprüft, ob Musk oder Trump einige dieser Kommunikationsstrategien anwenden.

Zuvor ist es jedoch hilfreich, sich zuerst mit den beiden Korpora vertraut zu machen. Dazu können in CQPweb ein paar Korpusqueries generiert werden, mithilfe derer der Kontext der Schlagwörter, die Verteilung über Monate, Jahre, Tage, die Verwendung in Bezug auf die beiden Plattformen Truth Social und Twitter, ob es sich um Retweets handelt und vieles mehr genauer betrachtet werden. In der ersten Tabelle werden 13 zufällige Ergebnisse der simpleQuery (case-insensitive) „Trump“ und in der zweiten Tabelle deren Verteilung über die Jahre 2009-2025 hinweg gezeigt.

Your query "Trump" returned 12,143 matches in 10,746 different texts (in 2,249,018 words [81,776 texts]; frequency: 5,399.25 instances per million words), ordered randomly

[0.175 seconds]

I< << >> >I Show Page: 1 Line View Show in corpus order Choose action... Go!

No.	Text	Solution 1 to 50 Page 1 / 243		
1	62455 2023-03-30	m Soros , has indicted a former U.S. President . This unprecedented attack against	Trump	is an assault on everything we hold sacred about our Republic . If they can come fo
2	68789 2023-11-13	RT @alxDonald	Trump	, Tucker Carlson , Kid Rock , Dana White and Don Jr. are at UFC tonight 🇺🇸
3	14716 2014-06-10	" " @anjeleyz : This is my nephew Mr	trump	, he is having chemo again , can you look at this and send a message maybe " " Ty
4	21610 2015-05-19	" " @lancebagley1 : Great Article , DonaldTrump ISIS in competition w/	Trump	# MakeAmericaGreatAgain & , Kill Islambies !!! http://t.co/69DeTcEfat " "
5	19103 2015-01-15	" " @ahansen20 : @realDonaldTrump	Trump	National Doral = Beatiful , amazing , & , # BestShowersEver ... and Gary Players pi
6	42619 2019-11-14	Mr. Kent affirmed Ukraine has a long - standing history of corruption . And President	Trump	was deeply skeptical of this cor ...
7	38675 2019-06-11	"	Trump	administration gives final approval for year - round E15 use " https://t.co/WPrDiyfpl
8	44076 2019-12-27	RT @TrumpWarRoom : WATCH : @ Franklin_Graham says President	Trump	" " has done so much for our country . " " His father Billy Graham " " would be very
9	08089 2013-05-12	" " @71 Kristine : @realDonaldTrump Thank you for Fund Anything Mr.	Trump	you know how to help people in all the right ways . With hope " "
10	54280 2020-10-20	" Stock Markets will hit new highs if President	Trump	wins . Tremendous growth like never before . If Biden wins , it 's strangulation . Not
11	24716 2015-10-29	" " @victoryorbust : The One , The Only , Donald	Trump	, will and can Make America Great Again ! @ Reince @blewthebigone https://t.co/5
12	77600 2024-08-14	BIG NIGHT for	Trump	Endorsed Candidates , including Tony Wied , who was given virtually no chance of
13	44768 2020-01-23	" The Democrats have now conceded that President	Trump	has not committed a crime . " @ AriFleischer
14	68259 2023-10-23	ow , the attack on Israel , which NEVER WOULD HAVE HAPPENED DURING THE	TRUMP	ADMINISTRATION . Our once great Country is a GIANT MESS . MAGA !!!

Based on classification: *year*

Category [↓]	Words in category	Hits in category	Dispersion (no. texts with 1+ hits)	Frequency [↓] per million words in category
2009	1,156	49	45 out of 56	42,387.54
2010	3,286	39	35 out of 142	11,868.53
2011	15,921	52	49 out of 773	3,266.13
2012	73,613	205	194 out of 3,531	2,784.83
2013	158,805	861	825 out of 8,144	5,421.74
2014	135,647	1,220	1,160 out of 5,784	8,993.93
2015	170,051	1,920	1,781 out of 7,536	11,290.73
2016	90,958	479	444 out of 4,225	5,266.17
2017	65,738	159	154 out of 2,602	2,418.69
2018	131,637	361	318 out of 3,568	2,742.39
2019	240,751	730	684 out of 7,818	3,032.18
2020	304,297	1,066	995 out of 12,236	3,503.16
2021	3,513	7	6 out of 158	1,992.60
2022	106,845	547	489 out of 3,517	5,119.57
2023	251,287	1,989	1,581 out of 8,224	7,915.25
2024	343,266	2,004	1,609 out of 9,041	5,838.04
2025	152,247	455	377 out of 4,421	2,988.56
Total:	2,249,018	12,143	10,746 out of 81,776	5,399.25

Abbildung 4 & 5: KWIC Ansicht und Verteilung der Treffer in Bezug auf die Jahre 2009-2025 mit Frequency und Dispersion

Mit einem Klick auf das blau gefärbte Suchwort kann der Kontext mitsamt POS-Tags untersucht werden. Bei dem Button *choose action* kann *Distribution* ausgewählt werden. So können die Ergebnisse verteilt über Jahre usw. betrachtet werden. CQPweb bietet jedoch eine Vielzahl weiterer nützlicher Funktionen, um eine schnelle Korpusanalyse zu ermöglichen. Da die Analyse jedoch nicht gut dokumentiert werden kann, erfolgt der Großteil der Analyse mit cwb-ccc und FlexiConc in drei verschiedenen Jupyter Notebooks.

4.1 Analyse mit CWB-CCC

Nun erfolgen einige allgemeine Informationen und Entdeckungen in Bezug auf die beiden Korpora TRUMP und MUSK. Dabei wurden beide Korpora mit cwb-ccc verglichen, was im Notebook 04 CWB_cwb-ccc.ipynb dokumentiert ist. Vergleicht man die Tokenanzahl und die Anzahl der Tweets von Trump mit denen von Musk, so fällt auf, dass das Korpus mit Trumps Tweets aus circa 87.000 Tweets , also mit insgesamt 2.249.018 Token besteht, während Musks Korpus etwa 55.000 Tweets mit 685.048 Token beinhaltet. Daraus folgt, dass ein Tweet bei Trump durchschnittlich 26 Token enthält, während ein Tweet bei Musk durchschnittlich 12,5 Token beinhaltet. Bei Musks Korpus befinden sich circa 100 abgeschnittene Tweets, die nicht korrekt gescrapt wurden. MUSK beinhaltet insgesamt 3.610 als Adjektiv getaggte Wörter, während TRUMP 5.603 Adjektive beinhaltet – natürlich sind hier sowohl fälschlicherweise als Adjektiv getaggte impliziert, während Adjektive, die falsch getaggt wurden, nicht dabei sind. Zu den 30 häufigsten Adjektiven bei MUSK zählen ausschließlich positiv konnotierte Wörter wie *true, good, great, many, more, real, next, other* und *high*. Bei TRUMP kommt das Lemma *great* mit großem Abstand am häufigsten vor (über 10.000 Mal), gefolgt von *more, many, new, other, big, fake, last, bad, republican, best, better*. Trump verwendet also nicht nur positiv konnotierte Adjektive. Unter seinen Top 30 sind auch häufig negativ konnotierte Wörter wie *fake, bad* oder *radical*. Für das Tagging der Korpora wurden zwei verschiedene Tagsets pos und upos verwendet. Vergleicht man beide Korpora in Bezug auf die Tagsets, so kommen ziemlich ähnliche Ergebnisse heraus. Verglichen wurden Adjektive und Adverbien, die bei pos mit ADJ und ADV und bei upos mit JJ und RB dargestellt werden. Upos unterscheidet zusätzlich außerdem JJS, JJR und RBR, RBS für Komparativ und Superlativ des Adjektivs oder Adverbs, sodass die Kategorien nicht identisch sind und dadurch nicht komplett übereinstimmen können. Über Trump ist bekannt, dass er gerne in Großbuchstaben schreibt und dadurch seine wichtigsten Slogans verbreitet. Neben MAKE AMERICA GREAT AGAIN postet er noch andere Slogans bzw. Bruchstücke von Slogans, wenn diese länger als vier Wörter waren: Auf der nächsten Seite sieht man links die häufigsten Treffer der Query '[word="[A-Z]+"]{4,}' bei TRUMP und rechts die Ergebnisse von MUSK. Auf den ersten Blick fällt auf, dass Trump deutlich häufiger in Großbuchstaben schreibt als Musk. Dadurch, dass die ersten Treffer zwischen 106 und 7 Mal im Korpus vorkommen, merkt man, dass Trump sich häufig wiederholt und einige Phrasen mehrfach postet. Musk hingegen scheint Großbuchstaben (zumindest in der Konstellation von vier großgeschriebenen Wörtern) nicht oft zu verwenden. Der Spruch, der bei Musk dreimal vorkommt ist „Vox populi vox dei“ und heißt auf deutsch: „Die Stimme des Volkes ist die Stimme Gottes“. Musks Slogan bezieht sich auf die Übernahme Twitters durch ihn und der damit verbunden Rückgabe der Macht an das Volk – wie er es zumindest schreibt (Dimri, 2024). Insgesamt kommen in Trumps Korpus 6.993 verschiedene großgeschriebene Wörter vor, während MUSK nur 922 verschiedene enthält – darunter bei beiden ganz vorne mit dabei „I“ und „RT“, was für Retweet steht. Darüber hinaus finden sich bei Musk vor allem Technik-spezifische Begriffe wie „AI“, „X“ oder „NASA“, während Trump (wie zu erwarten) hauptsächlich Worte postet, die unterstützen sollen, dass er gewählt wird. Die

häufigsten Wörter sind unter anderem „GREAT“,, „AMERICA“, „TRUMP“, „YOU“, „AGAIN“ und „MAKE“. Darunter ist eine Konstellation seines Slogans „MAKE AMERICA GREAT AGAIN“. Außerdem redet er oft von sich selbst oder seiner Familie, da „TRUMP“ das siebthäufigste Wort mit 1511 Vorkommnissen ist. Vermutlich zählt ein Großteil davon aber zu Retweets, die in Trumps Korpus sehr häufig vorkommen. Insgesamt sind 16.554 Tweets Retweets, wenn man der Zahl für „RT“ nach urteilt. Bei Musk kommt „RT“ 654 Mal vor und ist dabei trotzdem nach „I“ mit 5.007 Mal das zweithäufigste Wort in Großbuchstaben. Bei Trump taucht „I“ sogar 18.689 Mal auf, was sogar 8% aller Token des Korpus ausmacht.

item	freq
WILL NEVER LET YOU	106
WILL NOT LET YOU	68
WE WILL MAKE AMERICA	51
WILL MAKE AMERICA GREAT	39
UNITED STATES OF AMERICA	38
WORST PRESIDENT IN THE	24
TO MAKE AMERICA GREAT	20
TOO BIG TO RIG	19
WITCH HUNT OF ALL	13
WERE FOUR YEARS AGO	13
WOULD BE IMPOSSIBLE FOR	10
WE ARE GOING TO	10
WE ARE A NATION	9
WILL ONLY GET WORSE	8
WAS RIGHT ABOUT EVERYTHING	8
TRUMP DIGITAL TRADING CARDS	8
TRUMP WAS RIGHT ABOUT	8
TO THE UNITED STATES	7
WOULD NOT BE ABLE	7
WOULD NEVER HAVE HAPPENED	7

item	freq
VOX POPULI VOX DEI	3
AMERICA IS GOING TO	1
ARGENTINA TAKES THE LEAD	1
BILL MUST NOT PASS	1
AI ML LLM LSTM	1
C A R S	1
CRIMINAL BILL MUST NOT	1
GANADOR EN EL DEBATE	1
FREE TOMMY ROBINSON NOW	1
IS GOING TO MARS	1
KEEP VOTING NO LOOSE	1
KLAINER KEEP VOTING NO	1
I OFFERED THIS DIRECTLY	1
MATIAS GANADOR EN EL	1
ML LLM LSTM GPT	1
R R R R	1
PEOPLE SHARE THEIR EXPERIENCES	1
TAKES THE LEAD IN	1
TENEMOS LA MEJOR GRAMILLA	1
THE LEAD IN THE	1

Abbildungen 6 & 7: Die 20 häufigsten Slogans von Trump (links) und die 20 häufigsten Slogans von Musk (rechts)

Neben den eben beschriebenen allgemeinen Wortverteilungen in Bezug auf Adjektive, Adverbien und Großbuchstaben, ist eine weitere nützliche Funktion von cwb-ccc die Unterteilung in Subkorpora. Insgesamt wurden für Trumps Korpus sieben Subkorpora erstellt, die jeweils Tweets mit den Namen „Obama“, „Hillary“, „Biden“, „Kamala“, „Trump“, „China“ und „Russia“, um deren Kollokate zu vergleichen. CWB-ccc stellt für die Berechnung der Kollokate sehr viele verschiedene Algorithmen zur Verfügung, unter anderem „log-likelihood“ und „O2“. Diese beiden wurden in Bezug auf die so

entstandenen Kollokate und Keywords verglichen. In Verbindung mit „Trump“ tauchen am häufigsten die Wörter „Trump“, „Donald“, „President“, „Mr.“ und „Tower“ auf. In Verbindung mit Trumps politischen Gegnern erscheinen aussagekräftigere Wörter: Obama wird oft mit „President“ und „Barack“ kombiniert, aber auch mit „Administration“, „Syria“, „Iran“, „Ebola“, „ObamaCare“ und sogar „Putin“. Dies gibt uns Aufschluss über die Dinge, die Trump über Obama postet. Zum Beispiel waren die Kriege in Syrien und Iran und die amerikanische Beteiligung zu Obamas Amtszeit stark diskutiert. Im zweiten Jahr von Obamas Amtszeit wird nach langem Zögern ObamaCare, eine Art Krankenversicherung beschlossen (Wysmolek, 2024). Hillary Clinton wird vor allem in Kombination mit „Crooked“ und „beat“ geschrieben, was zeigt, wie Trump über andere Präsidentschaftskandidaten spricht. Auch Biden wird häufig mit „Sleepy“, „Crooked“ und „Hunter“ kombiniert. Bei Kamala Harris sind die Worte „Comrade“, „Lyin“ und „Debate“ stark vertreten. Harris steigt erst sehr spät in den Wahlkampf 2024 gegen Trump ein, nachdem Biden sich wegen seines hohen Alters und damit verbundenen Schwierigkeiten zurück zieht. Davor ist sie Vizepräsidentin von Biden und ist keine direkte Gegnerin von Trump. Kommen wir nun zu zwei Gegnern, die wohl eher die „Feinde von außen“ sind: China und Russland. China taucht primär mit „trade“, „Virus“, „tariff“, „deal“ und „dollar“ auf, was zeigt, dass es sich bei China vor allem um einen wirtschaftlichen Gegner handelt, bei dem Geld und Handel eine große Rolle spielt. Außerdem stammt das Corona Virus Covid-19, welches 2020/21 um sich greift, mutmaßlich aus China (dpa, 2025). Russland taucht häufig in Verbindung mit der Ukraine auf, was auf den Ukrainekrieg zurück geführt werden kann. Auch die Worte „hoax“, „collusion“, „Putin“, „Zelenskyy“, „FBI“ und „Intelligence“ sind Keywords des Subkorpus „Russia“. Zelenskyy ist der Präsident der Ukraine und damit in direkter Opposition zu Putin und Russland. Russland ist für seine weltweite Spionage und Falschmeldungen bekannt (Bundesamt für Verfassungsschutz, 2025), was sich auch in den Keywords widerspiegelt. Teilt man alle Tweets von Trump und Musk in Subkorpora ein, so fällt auf, dass die jeweiligen Keywords die politische Situation gut widerspiegeln. Trump beispielsweise nutzt den Begriff „fake“ in seiner ersten Amtszeit sehr häufig (1.282 Mal). Zur Amtszeit von Joe Biden kommt der Begriff knapp 700 Mal, obwohl das Subkorpus nur halb so groß ist wie das zu Trumps erster Amtszeit. In Trumps zweiter Amtszeit, also nur von Januar bis August 2025, befinden sich 2800 Posts (zum Vergleich: Im Subkorpus seiner ersten Amtszeit sind 30104 Texte, welche jedoch vier Jahre umfasst). Trump hat sein Postingverhalten also deutlich erhöht. Bei Musk kommt der Begriff „fake“ im Zeitraum von 2021-2024 84 Mal auf. Davor und danach nur circa 14 Mal.

4.2 Erarbeitung der Kommunikationsstrategien

In dieser Bachelorarbeit entscheide ich mich für den Begriff „Kommunikationsstrategien“, der zunächst wertneutral verstanden werden soll. Gemeint sind dabei im Rahmen der Bachelorarbeit vor allem im Folgenden spezifisch erklärte Strategien. Da diese meistens zur absichtlichen Manipulation oder Desinformation gebraucht werden, könnten diese Strategien auch als Desinformations- oder Manipulationsstrategien bezeichnet werden. Da ich auf keine der Optionen festlegen will, bleibt in dieser Bachelorarbeit der

neutralere Begriff „Kommunikationsstrategien“ bestehen. Das Paper „Detection of Persuasion Techniques in Texts and Images” von Dimitrov et al. (2021) nennt 22 verschiedene Kommunikationsstrategien (im engl. eigentl. “Persuasion techniques”), die ich im Folgenden vorstellen werde:

Die erste Strategie ist die der Benutzung **Geladener Sprache** (Loaded Language). Dabei geht es um den Gebrauch spezifischer Worte mit starkem emotionalen Einfluss oder Subtext. Unter die Verwendung **bestimmter Bezeichnungen oder Beleidigungen** (Name Calling or Labeling) fällt beispielsweise die Bezeichnung von Kamala Harris als *crazy Kamala*. Dadurch wird die 2024 gegen Trump kandidierende Präsidentin als verrückt abgestempelt. In Kombination mit häufiger Wiederholung dieser Phrasen bleiben diese eingängigen Bezeichnungen hängen und schädigen langfristig den Ruf der betroffenen Personen. Mit der Strategie des **Zweifels** (Doubt) werden bestimmte Aussagen oder wissenschaftliche Tatsachen gezielt in Frage gestellt. Meist wirken diese Zweifel suggestiv und beinhalten fragwürdige Unterstellungen. Mit der vierten Strategie der **Über- oder Untertreibung** (Exaggeration or Minimisation) erscheinen Themen als schlimmer oder irrelevanter als sie eigentlich sind. Diese Strategie wird auch in privaten Gesprächen oft unterbewusst und unabsichtlich eingesetzt. Mit Absicht eingesetzt kann diese Strategie jedoch starke Auswirkungen auf die Stimmung der Bevölkerung haben, wenn ein Thema besonders bedrohlich erscheint. Verbunden wird diese Strategie oft mit der **Verbreitung von Angst und Vorurteilen** (Appeal to Fear or Prejudice). Dabei wird die eigene Meinung oft durch das Schüren von Ängsten gegenüber gegnerischen Positionen gestärkt. Dies wird oft zudem durch Vorurteile verstärkt. Die Verwendung griffiger **Slogans** gehört wahrscheinlich zu jeder Partei oder Person, die sich zu einer politischen Wahl aufstellen lässt. Oft gehen diese Slogans mit emotionalen Implikationen oder stereotypischen Behauptungen einher und zeichnen sich durch Einprägsamkeit und Kompaktheit aus. Donald Trumps wohl bekanntester Slogan ist der Spruch „Make America great again“. Bei der Strategie des **Whataboutism** werden Gegenpositionen bewusst durch den Vorwurf der Heuchelei geschwächt ohne dabei auf die eigentlichen Argumente der Gegenseite ein zu gehen. Die achte Strategie, die Dimitrov et al. nennen, ist das „**Fahneschwenken**“ (Flag-Waving). Diese Strategie setzt auf das Wecken nationaler Gefühle oder positiver Gefühle in Bezug auf eine Ethnie, eine politische Präferenz, ein bestimmtes Geschlecht oder eine Gruppe, um bestimmte Ideen durchzusetzen. Sehr oft wird sich auch der **Strohmann-Strategie** (Straw Man) bedient, indem eine generische Position falsch dargestellt wird. Anschließend wird gegen die „neue“ Position argumentiert. In der Regel ist die Strohmann-Position schwächer als die der ursprünglichen Gegenseite. Was ebenfalls immer wieder strategisch genutzt wird, ist die **kausale Vereinfachung oder Monokausalität** (Causal Oversimplification) bestimmter Themen. Dabei wird eine scheinbar einfache Lösung für ein komplexes Problem vorgeschlagen. Alternativ oder ergänzend wird den vermeintlichen Verursachern Schuld zugewiesen, ohne selbst konkrete Lösungen vorzuschlagen. Unter der Verwendung des **Autoritätsarguments** (Appeal of Authority) versteht man einen logischen Fehlschluss, bei dem eine Behauptung allein dadurch begründet wird, dass besagte Behauptung von einer Autoritätsperson ausgesprochen wurde. Der Autor wird als Experte hingestellt und gegenüber anderen Personen aufgewertet. Stellt

man sich gegen die Aussage, so stellt man sich auch gegen die Autorität der Person. Umgekehrt kann auch Personen ihr Wissen abgesprochen werden, indem sie auf andere Bereiche ihrer Interessen reduziert oder ihre akademischen Titel weggelassen werden. Durch das Autoritätsargument entsteht in der Öffentlichkeit ein verzerrtes Bild tatsächlicher Situationen, da unwissenschaftliche Aussagen „in wissenschaftlicher Autoritätsverpackung" dargestellt werden (*Autoritätsargument*, n.d.). Die 12. Strategie ist die der **Gedankenlenkenden Klischees** (Thought-Terminating Cliché), welche kritisches Denken oder sinnvolle Diskussionen unterbinden, indem sie knappe Antworten zu komplexen Themen liefern oder vom Thema ablenken. Diese Strategie tritt auch in Verbindung mit der Strategie der Monokausalität auf. Beim **Schwarz-Weiß-Fehlschluss oder Diktatur** (Black-White-Fallacy or Dictatorship) werden zwei Optionen als einzige Möglichkeit aufgezeigt, obwohl mehr Handlungsmöglichkeiten existieren. Im Extremfall wird diktiert, was genau getan werden soll. Die nächste Kommunikationsstrategie führt alle Aussagen einer Gegenposition auf gehasste Gruppen zurück. Wenn jemand zum Beispiel als Nazi abgestempelt wird, ist es für ihn teilweise schwer, wieder rehabilitiert zu werden. Diese Strategie wird **Reductio ad Hitlerum** genannt. Die besonders einprägsame Strategie der **Wiederholung** (Repetition) sorgt dafür, dass Empfänger der entsprechenden Botschaften immer wieder an bestimmte Aussagen erinnert werden und sich durch häufige Wiederholung gut einprägen können, bis die Zielgruppe durch häufige Wiederholungen irgendwann möglicherweise eher zustimmt. In Verbindung mit einprägsamen Slogans können beispielsweise politische Meinungen kompakt transportiert werden. Durch häufige Wiederholungen bleiben diese besonders gut in Erinnerung. Unser menschliches Gehirn lernt durch Wiederholung und erhält dabei den Eindruck, dass die wiederholten Informationen besonders relevant sind. Diese Strategie wird unter anderem bei Werbung oder Algorithmen auf Social Media genutzt. Durch **Verschleierung, Verwirrung und Absichtliche Vagheit** (Obfuscation, Confusion, Intentional Vagueness) wird der Zielgruppe viel Spielraum für eigene Interpretationen gelassen, da entsprechende Thesen nur ungenau und vage geäußert werden. Denkt man an die Schriften verschiedener Religionen, so bestehen dort viele Unklarheiten, über die Menschen seit Jahrhunderten diskutieren und individuell auslegen, was das Geschriebene zu bedeuten hat. Beim **Ablenkungsmanöver** (Red Herring) werden für eine Diskussion irrelevante Argumente angeführt, um von der eigentlichen Argumentation abzulenken. Mit der Strategie des **Bandwagon** erfolgt der Versuch, eine Zielgruppe mit der Argumentation, dass alle anderen es auch tun zu bestimmten Handlungen zu überreden. Es ist quasi die Strategie des Gruppenzwangs. Bei **Verleumdung** (Smears) wird der Ruf anderer Personen geschädigt, indem negative Propaganda betrieben wird. Die Verwendung **glänzender Tugenden** (Glittering Generalities, Virtue) bedient sich wichtiger Symbole eines Wertsystems, um das eigene Ansehen zu steigern. Wenn sich jemand als besonders ehrlich hinstellt, so kann das sein Ansehen steigern. Die 21. Strategie der **Verwendung starker Emotionen** (Appeal to strong Emotions) beruft sich auf Bilder zum Wecken starker Emotionen, um Personen zu beeinflussen. Die letzte Strategie ist die der **Assoziation** oder des **Transfers** (Transfer or Association). Sie projiziert starke emotionale Reaktionen auf Personen, Themen oder Werte, um diese zu diskreditieren.

Die eben genannten Strategien sind in einigen Fällen nicht klar voneinander abgrenzbar, sondern überschneiden sich in manchen Aspekten. In den beiden folgenden Kapiteln werde ich prüfen, welcher Kommunikationsstrategien sich Trump und Musk bei ihren Tweets auf Twitter und Truth Social bedienten. Dabei werde ich nur ausgewählte Kommunikationsstrategien beschreiben, da die vollständige Darstellung der erfolgten Analyse den Rahmen sprengen würde. In den beiden Jupyter-Notebooks 05 und 06, welche im Anhang näher beschrieben werden und sich im Git Hub Repository „TrumMus" befinden, ist jedoch die vollständige Analyse aller Kommunikationsstrategien mit FlexiConc abgebildet. Es befinden sich etwa 88 verschiedene Queries mit durchschnittlich vier Knoten in den Notebooks. Bevor ich auf die Analyse genauer eingehen werde, erfolgt eine kurze Beschreibung zu den Personen Donald Trump und Elon Musk, um die entsprechenden Ergebnisse besser einordnen zu können.

4.3 Einordnung und Analyse von Trump mit FlexiConc

Donald Trump war ursprünglich Unternehmer und wendete sich ab 2011 immer mehr der Politik zu. 2016 wurde er dann zum Präsidenten gewählt und ersetzte so den zuvor zweimal als Präsident vereidigten Barack Obama. 2020 verlor Trump die Wahl gegen Joe Biden. Dieser kandidierte auch zur Präsidentschaftswahl 2024 erneut – ebenso wie Trump. Aufgrund starker Alterserscheinungen stieg Biden während der Wahl aus und wurde durch Kamala Harris, die vorherige Vizepräsidentin ersetzt. Schließlich gewann jedoch Donald Trump die Wahl 2024 und wurde somit zum zweiten Mal Präsident. Erwähnenswert ist außerdem, dass Trump der erste (Ex-)Präsident der Geschichte ist, der seinen Wahlverlust 2020 nicht anerkannte und mit dem Sturm aufs Kapitol im Januar 2021 in die Geschichte einging - auch wenn Trump seiner Aussage nach daran wohl nicht beteiligt gewesen sein soll. Zudem war auch die Wahl 2024 geschichtsträchtig. Donald Trump wurde in 34 Anklagepunkten wegen Dokumentenfälschung, Vertuschung durch Schweigegeldzahlungen und anderen Punkten für schuldig gesprochen. Da er nach US-Verfassung dennoch als Präsident vereidigt werden kann, hielt ihn das nicht von einer erneuten Kandidatur ab. Diverse Versprechungen wie etwa die Beendigung des Ukrainekriegs innerhalb von 24 Stunden nach Amtsantritt bleiben bloße Versprechungen. Nach der eben erfolgten knappen Einordnung Donald Trumps in das Zeitgeschehen Amerikas erfolgt nun die Analyse der Kommunikationsstrategien, die im vorherigen Kapitel erklärt wurden. Dabei werde ich mich auf ausgewählte Erkenntnisse beschränken, da die Seitenanzahl der Bachelorarbeit keine vollständige Beschreibung jeder Kommunikationsstrategie zulässt. Die vollständige Analyse findet sich in den beiden Jupyter-Notebooks *05 FlexiConc_Analyse* und *06 FlexiConc_Kommunikationsstrategien*, sowie weitere Analysen mit cwb-ccc im Notebook *04 CWB_cwb-ccc*.

Die eben beschriebenen Strategien werden nun bei Trumps Tweets gesucht, um zu sehen, ob und welche Kommunikationsstrategien Donald Trump verwendet. Für die Analyse wurden circa 88 verschiedene Queries ausgewählt, zu denen jeweils Konkordanzanalysen mit durchschnittlich vier Knoten erfolgten. Im Anhang befindet sich eine vollständige Liste aller Queries für jede Kommunikationsstrategie – auch wenn sich manche Ergebnisse der Queries mit anderen Kommunikationsstrategien überschneiden. Bei

den Knoten wurden hauptsächlich zufällige Beispiele, Kontextwörter links und rechts und nach Jahren sortierte Ergebnisse untersucht. Bei der Kommunikationsstrategie **Name Calling** wählte ich hauptsächlich politische Gegner Trumps im Vergleich zu Sympathisanten aus. Gleichzeitig geben die Erkenntnisse daraus Aufschluss über die Verwendung weiterer Kommunikationsstrategien. Barack Obama ist Trumps Vorgänger in Bezug auf die Präsidentschaft 2016, sodass er in erster Linie kein Gegner Trumps darstellt. Obamas maximale Legislaturperiode von acht Jahren ist bereits erreicht, sodass dieser 2016 nicht erneut kandidieren konnte. Von Obama wird in Trumps Tweets sehr respektvoll gesprochen. Die häufigste Konkordanz ist „president“ mit 278 Treffern. Im Gegensatz dazu stehen Hillary Clinton, gegen die er 2016 kandidiert, Joe Biden, gegen den Trump 2020 die Präsidentschaftswahl verliert und Kamala Harris, die 2024 Joe Biden bei der Präsidentschaftswahl ersetzt. Als Unterstützer kann Elon Musk gesehen werden, der sich jedoch erst sehr spät in die Politik an der Seite von Trump einmischt. Da die gesammelten Daten von 2009 bis 2025 reichen, ist Musk nicht stark repräsentiert. Auf weltpolitischer Ebene sind China und Russland Opponenten Amerikas. Schaut man sich nun die häufigsten Konkordanzen bezogen auf den linken und rechten Kontext der gesuchten Token an, so fällt stark auf, dass von Obama am häufigsten in Kombination mit seinem Vornamen Barack oder dem Attribut des Präsidenten gesprochen wird. Auch Kamala Harris, Joe Biden und Hillary Clinton treten am häufigsten in der Kombination von Vor- und Nachnamen auf. Jedoch werden Kamala, Joe und Hillary auffällig oft auch ohne ihren Nachnamen erwähnt. Joe (Biden) wird 751 Mal mit dem Adjektiv „crooked“, was umgangssprachlich so viel wie unehrlich, korrupt bedeutet, bezeichnet. Auch die Worte „sleepy“ (255 Mal), „stupid“ und „incompetent“ treten sehr häufig in Kombination mit Joe (Biden) auf. Hillary (Clinton) wird ebenfalls als „crooked“ (429 Mal) bezeichnet. Kamala (Harris) wird ebenso mit „crooked“ beleidigt, aber auch mit „crazy“, „defeat“ und „lyin“. Abgesehen davon sind häufige Konkordanzen mit „Kamala“ „Biden“, „comrade“, „migrant“, „broke it“, „inflation“, „weak“, „fake“ und „invasion“. Bei Kamala Harris wird also ihre teils migrantische Herkunft betont. Es heißt an einigen Stellen zum Beispiel „end Kamala’s invasion“, wie unter 5. **Verbreitung von Angst und Vorurteilen** durch die Query „[lemma="threat|crisis|attack|terror|danger|collapse|invasion|enemy|riot"]“ herausgefunden werden kann. Außerdem bezeichnet Trump Harris als schwach, falsch, verrückt, lügnerisch und korrupt. Donald Trump erweitert sein Vokabular an beleidigenden Adjektiven für Kamala Harris über die Jahre seiner politischen Karriere hinweg. Doch die polemisierenden Worte der Query unter 5. Werden prinzipiell auf so ziemlich jede Opposition Trumps angewandt. Seien es Länder wie Iran oder China, ObamaCare, die Medien oder Amerikaner wie Obama, Harris oder Biden, der 15 Mal als Bedrohung verkauft wird („Biden is a threat to democrats“).

Auch von **Verleumdungen** macht Trump regelmäßig Gebrauch. In Verbindung mit „crooked“, „sleepy“ oder „lyin“ und dem folgenden Namen werden oft Vorwürfe und Behauptungen wie (bei Biden) „the worst president ever“ oder „It only came to a stop when the economy slammed into a rock called Crooked Joe“ aufgestellt, die rufschädigend wirken. Der zweite Spruch gibt Joe Biden zusätzlich die Schuld an der schlecht laufenden Wirtschaft, was zu kurz gegriffen ist. Schon seit 2014 befindet sich in Amerika in einer

Wirtschaftskrise, aus der es bisher noch nicht aufwärts geht. Joe Biden ist dafür jedoch nicht verantwortlich. Trotzdem so zu argumentiert fällt unter die Strategie der **Monokausalität** und des **Transfers**. Für die Wirtschaftskrise gibt es mehrere komplexe Gründe. Nur Joe Biden als Grund zu nennen, ist also falsch. Mit dieser Aussage und anderen Verleumdungen werden negative Emotionen auf Biden übertragen, die ihn als Option für die nächste Wahl unattraktiv machen sollen, was auch die Kommunikationsstrategie des Transfers beschreibt. Der nächste Satz des eben genannten Beispiels ist „@realDonaldTrump talks [sic] first steps when he is back in the White House, like reclaiming American energy". Mit dieser Aussage schlägt Trump im Prinzip zwei Optionen vor, was zu der Strategie des **Schwarz-Weiß-Fehlschlusses** zählt. Zum einen können die Bürger einen korrupten Präsidenten (Biden) wählen, der die Wirtschaft stoppt oder sie entscheiden sich bei der nächsten Wahl für Trump, der gleichzeitig verspricht, die „amerikanische Energie" zurück zu bringen. Die amerikanische Energie steht wohl für nationalen Spirit, auf den man stolz sein kann, was zu der Strategie **Flag-Waving** passt. Beim Flag-Waving werden nämlich nationale, positiv konnotierte Gefühle geweckt, zu denen die amerikanische Energie auch passt. Die eben beschriebenen Stellen finden sich bei der Query „Joe" und deren ersten Knoten „j1" unter *2. Name Calling* im Notebook 05.

```
In [21]:
1 j1.root.add_arrangement_node(
2 grouping = ("Partition by Ngrams",
3     {'positions': [-1]})
4
5 lay(HTML(generate_analysis_tree_html(joe)))
6 lay(HTML(generate_concordance_html(joe, j2, n=1000, metadata_columns=
```

A GREAT AGAIN! Crooked	Joe	Biden, who is killing the United Autoworkers with his WEAK stance on China and his ridiculous insistence on All
into a rock called Crooked	Joe	. RT @realamericasvoicePresident @realDonaldTrump talks first steps when he is back in the White House, like reclaiming American energy
s Biden betrayal. Crooked	Joe	should be ashamed to show his face before these hardworking Americans he is stabbing in the back. With Biden
rder to "bail out " Crooked	Joe	Biden, THE WORST & MOST CORRUPT PRESIDENT IN THE HISTORY OF THE UNITED STATES. The Old Crow,
'm also up 10 on Crooked	Joe	! What is the RNC doing? They should be fighting against Election Interference & the Pennsylvania Voter Registration Scam
'm also up 10 on Crooked	Joe	! What is the RNC doing? They should be fighting against Election Interference & the Pennsylvania Voter Registration Scam
-trump-in-ottumwa-ia.html Crooked	Joe	Biden has three major problems, and they all begin with I - INFLATION, IMMIGRATION, & INCOMPETENCE!
tupid people, like Crooked	Joe	Biden, are making the dumbest of all decisions for our once great Country. WE NEED SMART PEOPLE,

Besonders erwähnenswert ist zudem die Kommunikationsstrategie von **Slogans**. Neben dem bekannten Slogan „Make America great again" in diversen Abwandlungen ist mir die Redewendung „someone ate/is eating someone's lunch" aufgefallen. Diese Redewendung meint, dass die erste Person deutlich angsteinflößender und stärker ist als die Person, die gegessen wird. Der Spruch ist eine Metapher, die bildlich veranschaulicht, dass jemand besiegt oder überlistet wird. Auf dieses Sprichwort in Trumps Tweets stieß ich zufällig aufgrund der Query „Russia". Putin und China scheinen „uns" wohl zu überlisten, wenn nichts

dagegen unternommen wird, schenkt man Trump Glauben. „Putin has eaten Obama's lunch" oder „China is eating tariffs/our lunch" sind Metaphern, die sich jeder gut vorstellen kann und durch die Bildersprache besonders leicht visuell veranschaulicht werden können. Metaphern bleiben gut im Gedächtnis. Doch auch andere bildliche Formulierungen mit dem Wort „eat" grenzen fast an Humor: „Terrorists pouring in, Inflation eating our hearts out – WHAT THE HELL DO YOU [DO]?", wecken aber gleichzeitig tiefsitzende Emotionen und existentielle Sorgen. Das **Wecken starker Emotionen** in Kombination mit dem **Schüren existenzieller Ängste** zählt ebenfalls zu den in Kapitel 4.2 beschriebenen Kommunikationsstrategien. Sind erst einmal grundlegende Emotionen geweckt, so lässt sich der Diskurs nur schwer mit logischen Argumenten gewinnen.

Russland und China werden in Verbindung mit stereotypischen Bezeichnungen wie bereits im Kapitel zu cwb-ccc anhand der Keywords beschrieben, in Trumps Tweets erwähnt. Dabei handelt es sich um die Kommunikationsstrategie der **Gedankenlenkenden Klischees**. Russland erscheint häufig in Verbindung mit Hexenjagd („Witch Hunt") oder Spionage im weitesten Sinne. China hingegen erscheint oft als Konkordanz von Russland oder in Kombination mit dem Wortfeld stehlen, Geld, Jobs, Wirtschaft und Handel. Im linken Kontext steht an erster Stelle Präsident Xi von China, dicht gefolgt von „deal with". Diese Worte führen dazu, dass China als Handelspartner gesehen wird, mit dem geschickt umgegangen werden muss, um ebenfalls Gewinne zu erhalten. Russland wird bei Trumps Tweets sehr oft dreimal wiederholt, was wie das Kopfschütteln oder Tadeln eines Erwachsenen gegenüber Kindern wirkt. Russland ist bekannt für Spionage (Spying als Konkordanz), die Verbreitung von Falschmeldungen (hoax), was Trump nutzt, um das Feindbild zu stärken. Außerdem wehrt sich Trump mehrfach gegen den Vorwurf, dass er geheime Absprachen mit Russland treffen würde: „[..] no collution with Russia (so ridiculous!)". Ansonsten spricht Trump knapp 200 Mal von Gesprächen mit Putin („meeting/conversation/met with president Putin"), was auch der einzige Kontext ist, in dem von Putin gesprochen wird. Im Gegensatz dazu wird das amerikanische Volk an verschiedenen Stellen aufgewertet und mithilfe nationaler Gefühle verbunden, was zu der Kommunikation des **Fahne-Schwenkens** passt.

Bei jeder der bisher beschriebenen Kommunikationsstrategien fällt auf, dass Trump sich sehr oft wiederholt und der nahezu identische Wortlaut mehrfach vor kommt. Dazu zählen vor allem Beleidigungen und Verleumdungen wie der Satzteil „[Biden] is the worst president ever/ in the history of America / the United States oder andere Abwandlungen der Aussage, Biden sei der schlechteste Präsident aller Zeiten. Auch klischeebezogene Aussagen wie die Hexenjagd in Verbindung mit Russland oder Diebstahl in Verbindung mit China werden mehrfach in fast derselben Formulierung gepostet, ebenso wie Beispiele für gedankenlenkende Klischees (no collution with Russia,..). Durch **Wiederholungen** bleiben Trumps Aussagen im Gedächtnis. In Kombination mit prägnanten Formulierungen wie auch seine Slogans „Make America great again" oder „someone (Putin/Russia/They) is eating/ ate our/Obama's lunch" ist sofort klar, welchen Standpunkt er vertritt. Durch taktische Wiederholungen können die getroffenen Aussagen quasi nie wieder vergessen werden und landen im Langzeitgedächtnis. Schaut man sich die zeitliche

Abfolge der Wiederholungen an, so fällt auf, dass diese mehrfach innerhalb eines Tages und der darauf folgenden Woche geteilt werden. Anschließend wird der Slogan teilweise sogar Jahre später wieder aufgegriffen. Das Phänomen erinnert mich an den Schulunterricht, nach dem wir Englischvokabeln oder Geschichtseinträge dreimal innerhalb kürzester Zeit wiederholen sollten, um das Wissen im Langzeitgedächtnis zu behalten und auch Jahre später noch an den Unterrichtsstoff erinnern zu können. Genau dieses Mechanismus bedient sich Trump auch mit seinen Slogans und bei anderen Aussagen, die sich nicht auf der sachlichen Ebene abspielen.

4.4 Einordnung und Analyse von Musk mit FlexiConc

Elon Musk gehört zu den reichsten Menschen der Welt und ist seit 1995 Unternehmer. Bekannt wurde er hauptsächlich durch die Unternehmen Tesla, Neuralink und SpaceX. Seine politische Meinung war zunächst wohl liberal, auch wenn er sich nicht in eine politische Richtung einordnen wollte. Beispielsweise spendete er für den Präsidentschaftswahlkampf von Obama und Clinton. Ab der Corona-Pandemie 2020 änderten sich seine politischen Einstellungen nach und nach zu rechtspopulistischen Ansichten. Im Jahr 2022 erwarb er 80% der Anteile an Twitter und übernahm damit die Kontrolle über den Micro-blogging Dienst. Nach eigener Aussage strebte er nach einer besseren Gewährleistung von Redefreiheit (Dimri, 2024). Somit werden seitdem auch Beiträge, die vorher als diskriminierend, verhetzend oder Falschinformationen verbreitend aufgefasst und gelöscht wurden, nicht mehr gesondert behandelt. Auf der anderen Seite droht Musk beispielsweise anti-zionistischen Nutzern mit Löschung der Kommentare. Im Jahr 2023 wird Twitter in „X“ umbenannt (Arun et al., 2024). Außerdem nutzt Musk Twitter seitdem, um seine Reichweite zu erhöhen und seine politischen Ansichten zu erhöhen. Ab dem Sommer 2024 unterstützte Musk Donald Trump und engagierte sich auch in Europa für rechtspopulistische Parteien (wie etwa die AFD in Deutschland) und rechtsextreme Bewegungen. Für 130 Tage wirkte Musk mutmaßlich an der „Leitung der neu eingerichteten Organisation *Department of Government Efficiency (DOGE)* der US-Regierung“ mit, woraufhin es unter anderem zu radikalem Stellenabbau im Staatsdienst kam. Ende Mai 2025 gab Musk den Posten an der Seite von Trump auf. Was den Ukrainekrieg angeht, so ist Elon Musk maßgeblich beteiligt, indem er unter diversen Bedingungen Satelliten von SpaceX zur Verfügung stellt, um Kommunikation und Kriegsplanung im Land zu ermöglichen (*Elon Musk*, 2025).

Welche Kommunikationsstrategien wendet Musk nun an? Im Gegensatz zu Trumps Tweets enthalten Musks Tweets keinerlei Name Calling gegenüber Obama, Biden, Clinton oder Harris in Form von Adjektiven. Allerdings unterstellt er Joe Biden, einen nicht binären Tiefenstaat begründen zu wollen und schreibt 2025 deshalb: „I didn’t vote for Joe Biden’s non-binary deep state theater“. Diese unterstellt Biden, zu den Verschwörungstheoretikern der Deep State Theory anzugehören und **labelt** Biden als Verschwörungstheoretiker. Außerdem schwingt bei diesem Satz mit, dass Musk nicht viel von LGBTQ hält. Der Tweet „The Biden puppet is lying.” Zählt zu der Kommunikationsstrategie der **Verleumdung** oder Unterstellung, da Musk behauptet, Biden sei eine bloße Puppe, die lügt. Insgesamt kommt „Biden“ jedoch

nur 138 Mal in Musks Tweets vor und ist somit bei 55.000 Tweets bzw. 685.048 Token, was verschwindend gering ist. Auch Kamala (Harris) wird bei Musk als Puppe bezeichnet. Musk behauptet außerdem, dass Harris ein Desaster wäre, während Trump gleichzeitig als Retter der Demokratie hingestellt wird: „@realDonaldTrump Trump will safe democracy and America. Mark my words. Kamala / Harris would be a disaster." Dies lässt dem Leser und Wähler zwei Optionen offen, von denen eine der beiden Optionen als unattraktiv dargestellt wird. Diese Strategie ist die des **Schwarz-Weiß-Fehlschluss**. Allerdings wird Kamala (Harris) insgesamt nur etwa 20 Mal unterstellt, sie sei eine Puppe und lediglich 52 Mal erscheint ihr Name überhaupt in Musks Tweets, was in keinem Verhältnis zu Trumps Tweets steht. Auffällig ist hier jedoch, dass der Name „Harris" nur 15 Mal erwähnt wird und sich dabei auf Kamala Harris bezieht. Ihr Nachname erscheint fast ausschließlich in Kombination mit Biden. Bei der Query [lemma="threat|crisis|attack|terror|danger|collapse|invasion|enemy"] erscheinen die gesuchten Begriffe hauptsächlich in Kombination mit Redefreiheit („This is a massive attack against freedom of speech.") oder Pronatalismus, also die Auffassung, dass viele Kinder geboren werden sollten, damit die Zivilisation fortbestehen kann: „Population collapse is a massive threat to civilization", „Doing my best to help the underpopulation crisis. A collapsing birth rate is the biggest danger civilization faces by far." Musk schürt hier Angst vor dem Untergang der Zivilisation (**Verbreitung von Angst und Vorurteilen**) und bietet dabei eine simple Lösung an, indem zivilisierte Menschen möglichst viele Kinder zeugen sollten (**Monokausalität**). Insgesamt ergab die Suche 370 Treffer, von denen sich einige außerdem auf sein Unternehmen „Tesla" beziehen: „And why aim your attacks at Tesla, when we're doing more than any other company on earth to advance green, sustainable energy". Insgesamt lässt sich auch anhand der in Kapitel 4.1 beschriebenen Schlüsselwörter feststellen, dass Musk als Unternehmer hauptsächlich Inhalte zu Star Link, Tesla und anderen unternehmensbezogenen Inhalten postet. Dabei schreibt er außerdem über politische Themen, was bisher jedoch nicht so sehr ins Gewicht fällt. Im Gegensatz zu Trump verfügt Musk nicht über eine derart ausgefeilte Bandbreite an Kommunikationsstrategien, sondern stellt lediglich Behauptungen auf, ohne den Leser (meistens) linguistisch zu manipulieren. Kommen wir zum letzten Punkt von Musks Tweets, der in diesem Kapitel beschrieben wird, nämlich Musks Einstellung zum Thema LGBTQ. Wie sich bereits im ersten Zitat dieses Kapitels gezeigt, schreibt Musk, dass er Bidens nicht-binären Tiefenstaat nicht gewählt habe. Mit Musks Aversion gegen LGBTQ (Arun et al., 2024) geht der Account „@EndWokeness" einher, von dem Musk 229 Mal schreibt. Der Suchbegriff „*woke*"(simple query) erscheint insgesamt 515 Mal in Musks Korpus. Als „woke mind virus" bezeichnet Musk sinnbildlich eine Krankheit, mit der angeblich eine Gruppe bestimmter Leute infiziert seien, die „wachsam" gegenüber sozialen Ungerechtigkeiten wie Rassismus oder Sexismus sein wollen. Der Begriff „woke" oder „wokeness" stammt aus dem afroamerikanischen Englisch und drückt ein Bewusstsein gegenüber sozialer, sexistischer oder rassistischer Unterdrückung aus (Kempter, 2024). Musk gehört der eher konservativen Gruppe an Leuten an, die dieser Strömung kritisch gegenüber stehen. In Kombination mit der Erwähnung „@EndWokeness" schreibt Musk, dass es verrückt sei, Raub sei, von der „woke Stasi" betrieben werde, oder dass das Virus traditionelle Medien übernehme („It's crazy; Robbery in LA is out of control; NPR

is run by woke Stasi; This is insane !!“) Außerdem bezeichnet Musk die Bewegung als illegal. Der Begriff „illegal“ ist Teil der **geladenen Sprache**, da er gerade in Amerika stark mit illegalen Einwanderern und Unrecht konnotiert ist, was eine starke emotionale Reaktion bei den Lesern hervorruft. Der Reim „go woke, go broke“ ist besonders eingängig und stellt einen **Slogan** gegen die Woke-Bewegung dar.

5. Fazit

Abschließend folgt nun das Fazit zu den in der Bachelorarbeit behandelten Themen. Dabei wird im Kapitel *5.1 FlexiConc* vor allem die Funktionsweise des Tools beschrieben und wie sich die Verwendung von FlexiConc im Hinblick auf die Themenstellung der Bachelorarbeit eignete. Anschließend folgt ein Fazit zu den Ergebnissen der Analyse der Social-Media-Kommunikationsstrategien. Abgerundet wird die Bachelorarbeit von einen Ausblick, was in weiterer Forschung von Interesse wäre.

5.1 FlexiConc

Bei FlexiConc handelt es sich um ein neues Tool, das für die Analyse von Konkordanzen sehr hilfreich ist. Aufgrund der verschiedenen implementierten Algorithmen kann strukturiert geforscht werden. Es gibt verschiedene Algorithmen, mithilfe derer Ergebnisse von Queries sortiert, neu strukturiert und ausgewählt werden können. In der Bachelorarbeit verwendete ich vor allem den Algorithmus „Random Sample“, um zufällige Ergebnisse stichprobenartig zu untersuchen. In nächsten Schritt kann man sich mit dem Algorithmus „Partition by Ngrams“ den linken oder rechten Kontext der Diskurseme anzeigen lassen. Optional kann man sich neben dem Diskursem mit Kontext in KWIC auch optional Metadaten anzeigen lassen, um die Ergebnisse der Analyse besser einordnen zu können. Nutzt man den Algorithmus „Partition by Metadata Attributes“, werden die Ergebnisse nach den gewählten Metadaten sortiert. Zum Beispiel kann man hier das Jahr als Attribut wählen und so vergleichen, wann und in welchem Zeitraum die meisten Ergebnisse der Query erfolgten. Da die häufigsten Kontextwörter oder die Jahre mit den meisten Ergebnissen zuerst angezeigt werden, sieht man auf einen Blick, welche Ergebnisse relevant sind. Anhand des implementierten Analysistrees wird jeder Schritt (Partition, Sort,..) genau dokumentiert und kann so später nachvollzogen werden. Im Analysistree wird bei jedem Schritt zudem die Anzahl der Treffer angezeigt, um den Überblick zu behalten. Ein gutes Verständnis der Funktionsweise der Algorithmen ist nicht erforderlich, da die Algorithmen auch so verglichen und getestet werden können. Natürlich sollte verstanden werden, was die jeweiligen Algorithmen mit sich bringen. Das Verständnis dieser wird durch die Dokumentation auf Git hub (Dykes et al., 2025a) und das Jupyter Notebook (Evert, 2025) ermöglicht. Hier werden allerdings bisher noch nicht alle Algorithmen erklärt und der Algorithmus „Partition with OpenAI“ kann beispielsweise nur mit einem bezahlten Account bei ChatGPT verwendet werden.

Der implementierte Analysistree für jede Query ermöglicht Transparenz in einem Maß, das bisher bei CADS noch nicht möglich war. Es wäre eine Überlegung wert, den Analysistree auch für andere

linguistische Untersuchungen wie Frequenzlisten, Kollokationen oder die Bildungen von Subkorpora zu implementieren, sodass transparenter dokumentiert werden kann. FlexiConc kann gut mit anderen Korpustools kombiniert werden, um eine umfassende Analyse zu ermöglichen. In Kombination mit CQPWeb und cwb-ccc oder auch MMDA ist FlexiConc eine wertvolle Ergänzung und Schnittstelle, die unbedingt beachtet werden sollte. FlexiConc ist nicht darauf ausgerichtet, eine weitere App oder ein weiteres Analysetool wie AntConc oder LancsBox zu sein, sondern stellt eine Ergänzung zu anderen Korpustools dar, da es Schnittstellen für CWB, CLiC, SketchEngine, CQPweb und andere Korpustools bietet. Will man keines der genannten Tools verwenden, können man auch einfache TSV-files für die Analyse mit FlexiConc verwenden. FlexiConc ist folglich sehr flexibel einsetzbar und ergänzt existente Korpustools aufgrund seiner Vielzahl an Algorithmen sinnvoll. Allerdings wäre es hilfreich, die Dokumentation mit weiteren Beispielen und Erklärungen zu den einzelnen Algorithmen und deren Funktionsweise auszubauen (was sich aktuell noch in Bearbeitung befindet). Insgesamt ist die Installation und Implementierung von FlexiConc mit den bisher verfügbaren Tutorials, die allesamt im Jupyter-Notebook 04 verlinkt sind machbar und verständlich. Allerdings ist die Kombination von FlexiConc und Jupyter Notebook ab einer bestimmten Menge an Queries mit durchschnittlich vier Knoten nicht unbedingt ratsam, da Jupyter Notebook seine Kapazität erreicht. Ich habe in den beiden Notebooks 05 und 06 insgesamt circa 88 Queries, womit Jupyter Notebook sehr zu kämpfen hat. Im Vergleich zu FlexiConc ist die Analyse über CQPweb deutlich schneller. Allerdings fehlt zum Beispiel die Sortierung nach Kontextwörtern, die ich persönlich sehr praktisch finde. Wenn die Dokumentation für FlexiConc noch weiter ausgebaut und mit Beispielen unterfüttert wird, kann ich mir gut vorstellen, dass FlexiConc sowohl für Linguisten, die nicht viel von Python verstehen als auch für Computerlinguisten und Spezialisten eine sinnvolle Ergänzung zu bisher vorhandenen Tools werden wird.

5.2 Fazit zur Analyse der Kommunikationsstrategien

Die in Kapitel 4 erfolgte Analyse der Kommunikationsstrategien über Konkordanzen mittels FlexiConc und stellenweise Ergänzungen durch CQPweb und cwb-ccc brachte definitiv interessante Ergebnisse zu Trumps und Musks Verwendung der Strategien hervor. Jedoch können lexikalische Ausdrücke oder grammatikalische Motive nicht immer Aufschluss über Kommunikationsstrategien bieten, da manche Strategien nur auf inhaltlicher Ebene erkannt werden können. Aus diesem Grund ist es schwierig, eine umfassende Analyse dieser anzubieten. Ergänzende Analysen durch Sentimentanalysen oder andere Methodiken kann dabei hilfreich sein. Ich habe mich allerdings bemüht, Kommunikationsstrategien zu wählen, die auf lexikalischer Ebene gefunden werden können. Abgesehen davon konnten auch erfolgte Queries bereits spannende Informationen über die Verwendung diverser Kommunikationsstrategien mitliefern, auch wenn die Analyse keineswegs den Anspruch der Vollständigkeit hat. Durch die quantitative Analyse über Stichwörter und deren Kontext können jedoch trotzdem aussagekräftige Ergebnisse erreicht werden.

In der bisher erfolgten Analyse fällt jedenfalls auf, dass Trump fast alle 22 Kommunikationsstrategien verwendet, während Musk kaum auf Kommunikationsstrategien zurückgreift. Insgesamt fällt auf, dass Musk selten auf persönlicher Ebene beleidigend wird. Außerdem können verschiedene Kommunikationsstrategien schwieriger anhand lexikalischer Begriffe festgestellt werden, da Musk hauptsächlich inhaltlich argumentiert. Elon Musk versucht nicht so sehr, seine Leserschaft von seinen Ansichten zu überzeugen, sondern äußert Behauptungen, ohne auf die Zustimmung oder Unterstützung vieler Leute angewiesen zu sein. Trump hingegen versucht mit verschiedenen Taktiken – wie in Kapitel 4.3 beschrieben -, die Bevölkerung und Wählerschaft von sich zu überzeugen, da er mindestens in Zeiten von Präsidentschaftswahlen darauf angewiesen ist, möglichst viele Stimmen zu akquirieren. Dafür greift er zu allen Mitteln und wendet nahezu alle Kommunikationsstrategien an, die in Kapitel 4.2 beschrieben wurden.

5.3 Ausblick

Das Feld der Datenbeschaffung über Daten aus dem Internet sollte an Bekanntheit in der Computerlinguistik gewinnen, da es heutzutage schwierig geworden ist, Daten auf anderen Wegen zu erhalten. Insbesondere die Verwendung von Playwright könnte deutlich bekannter werden. Wichtig ist dabei natürlich, dass bestehende Datenschutzverordnungen beachtet werden und keine Inhalte wiederrechtlich gescrapet werden. In zukünftiger Forschung wäre es zudem interessant, die beiden Datensätze TRUMP und MUSK um die neuesten Posts zu ergänzen und mithilfe besserer Tagger bzw. aus der Kombination verschiedener Modelle für Tokenisierung, Lemmatisierung und Tagging die besten Tagging-Ergebnisse zu erreichen. Aufgrund der zeitlichen Begrenzung der Bachelorarbeit von drei Monaten und der aufwändigen Datenbeschaffung, wurden nur vortrainierte Tagging-Modelle getestet und verglichen. Auch die Analyse der Tweets von Trump und Musk verdient größere Beachtung und umfangreichere Forschung als es in dieser Bachelorarbeit möglich war. Die beiden Personen bieten hervorragende Datenquellen, um daran zu forschen, wie Kommunikationsstrategien (Desinformationsstrategien) zielsicher erkannt und über linguistische Merkmale beschrieben werden können. Auch politische Anspielungen auf historische und politische Ereignisse durch Trump und Musk wären interessant. Manche rhetorischen Strategien wären außerdem im Kontext von Interviews oder Talkshows interessant, da dort Verhaltensweisen wie ständiges Unterbrechen, Ablenken vom Thema oder andere Strategien besser analysiert werden können.

6. Literaturverzeichnis

Abodayeh, A., Hejazi, R., Najjar, W., Shihadeh, L., & Latif, R. (2023). Web scraping for data analytics: A beautifulsoup implementation. *2023 Sixth International Conference of Women in Data Science at Prince Sultan University (WiDS PSU)*, 65–69.

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., & Vollgraf, R. (2019). FLAIR: An easy-to-use framework for state-of-the-art NLP. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, 54–59.

Almabruk, S., Abdalhamid, S., & Almabruk, T. (2025). Comparative Reliability Analysis of Selenium and Playwright: Evaluating Automated Software Testing Tools. *Asian J. Res. Comput. Sci*, *18*, 34–44.

Anthony, L. (2005). AntConc: Design and development of a freeware corpus analysis toolkit for the technical writing classroom. *IPCC 2005. Proceedings. International Professional Communication Conference, 2005.*, 729–737. https://doi.org/10.1109/IPCC.2005.1494244

Anthony, L. (2014). *AntConc (Windows, Macintosh OS X, and Linux)*.

Arun, A., Chhatani, S., An, J., & Kumaraguru, P. (2024). X-posing Free Speech: Examining the Impact of Moderation Relaxation on Online Social Networks. *Proceedings of The 8th Workshop on Online Abuse and Harms (WOAH)*, 201–211.

Autoritätsargument. (n.d.). Der Goldene Aluhut.

Baker, P. (2006). *Using Corpora in Discurse Analysis*. Continuum.

Baker, P., Gabrielatos, C., Khosravinik, M., Krzyżanowski, M., McEnery, T., & Wodak, R. (2008). A useful methodological synergy? Combining critical discourse analysis and corpus linguistics to examine discourses of refugees and asylum seekers in the UK press. *Discourse & Society*, *19*(3), 273–306.

Baker, P., & Levon, E. (2015). Picking the right cherries? A comparison of corpus-based and qualitative analyses of news articles about masculinity. *Discourse & Communication*, *9*(2), 221–236.

Baker, P., & McEnery, T. (2005). *A corpus-based approach to discourses of refugees and asylum seekers in UN and newspaper texts. Journal of Language and Politics 4*, 197–226.

Bale, A. S., Ghorpade, N., Kamalesh, S., BS, R., & others. (2022). Web scraping approaches and their performance on modern websites. *2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC)*, 956–959.

Bansal, M., DAR, M. A., & Bhat, M. M. (2023). Data Ingestion and Processing using Playwright. *Authorea Preprints*.

Barbaresi, A. (2021). Trafilatura: A web scraping library and command-line tool for text discovery and extraction. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, 122–131.

Boleda, G., Bott, S., Meza, R., Castillo, C., Badia, T., & López, V. (2006). CUCWeb: A Catalan corpus built from the Web. *Proceedings of the 2nd International Workshop on Web as Corpus*.

Brants, S., Dipper, S., Eisenberg, P., Hansen-Schirra, S., König, E., Lezius, W., Rohrer, C., Smith, G., & Uszkoreit, H. (2004). TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation*, *2*(4), 597–620.

Brown, B., & Williams, A. (2014). *Twitter Scraper* [Computer software]. https://github.com/bpb27/twitter_scraping

Brown, B., & Williams, A. (2024). *Trump Twitter Archiv V2*. TTA. https://www.thetrumparchive.com

Bundesamt für Verfassungsschutz. (2025, April). *Gefährdung durch russische Spionage, Sabotage und Desinformation*. Bundesamt Für Verfassungsschutz. https://www.verfassungsschutz.de/SharedDocs/hintergruende/DE/spionage-und-proliferationsabwehr/gefaehrdung-russische-spionage-sabotage-desinformation.html

Burnard, L., & Dodd, T. (2003). Xara: An XML aware tool for corpus searching. *Proceedings of Corpus Linguistics*, 142–144.

Derczynski, L., Ritter, A., Clark, S., & Bontcheva, K. (2013). Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. *Proceedings of the International Conference Recent Advances in Natural Language Processing Ranlp 2013*, 198–206.

Dimitrov, D., Ali, B. B., Shaar, S., Alam, F., Silvestri, F., Firooz, H., Nakov, P., & Martino, G. D. S. (2021). SemEval-2021 task 6: Detection of persuasion techniques in texts and images. *Proceedings of the 15th International Workshop on Semantik Evaluation*. SemEval-2021, Bankok, Thailand.

Dimri, S. (2024). *The Bird is Freed? A Discourse Analysis of The Populist Brand of Elon Musk*. https://doi.org/10.5281/zenodo.14178746

Dozat, T., & Manning, C. D. (2016). Deep biaffine attention for neural dependency parsing. *arXiv Preprint arXiv:1611.01734*.

dpa. (2025, June 30). *WHO-Rat: China mauert, Ursprung des Coronavirus unklar*. Deutsches Ärzteblatt. https://www.aerzteblatt.de/news/who-rat-china-mauert-ursprung-des-coronavirus-unklar-024dc24c-9e32-401b-98f6-86b53ca231e8

Dykes, N., Evert, S., Mahlberg, M., & Piperski, A. (2025a). *FlexiConc* (Version 0.1.5) [Computer software]. Friedrich-Alexander-Universität Erlangen-Nürnberg. https://github.com/fau-klue/flexiconc-docs

Dykes, N., Evert, S., Mahlberg, M., & Piperski, A. (2025b). *RC21 Tutorial FlexiConc* [Computer software]. https://github.com/reading-concordances/teaching/tree/main/tutorials/konvens2025

El Asikri, M., Knit, S., & Chaib, H. (2020). Using web scraping in a knowledge environment to build ontologies using python and scrapy. *European Journal of Molecular & Clinical Medicine*, *7*(03), 2020.

Elon Musk. (2025, October 24). Elon Musk. https://de.wikipedia.org/wiki/Elon_Musk#Politisches_Wirken

Elon Musks Vermögen erreicht 500 Milliarden Dollar. (2025, October 2). Tagesschau24 Live. https://www.tagesschau.de/wirtschaft/unternehmen/musk-500-milliarden-dollar-reichtum-tesla-100.html

Evert, S. (2008). 58. Corpora and Collocations. In *Corpus Linguistics* (Vol. 2, pp. 1212–1248). De Gruyter Mouton.

Evert, S. (2022). Measuring Keyness. *Digital Humanities 2022*, 202–205.

Evert, S. (2025). *FlexiConc 0.1.5—Demo Notebook*. https://colab.research.google.com/drive/1Y-yXs0RfDWU96Ex-aEYtMCetUtN20Tek?usp=sharing

Evert, S., & Hardie, A. (2011). Twenty-first century corpus workbench: Updating a query architecture for the new millennium. *Proceedings of the Corpus Linguistics 2011 Conference*, 1–21.

Evert, S., & Heinrich, P. (2019). *Introducing MMDA: An interactive toolkit for CDA*.

Evert, S., & The CWB Development Team. (2022). *The IMS Open Corpus Workbench (CWB) CQP Interface and Query Language Tutorial*.

Fairclough, N. (2015). *Language and Power* (3 edition).

Farrow, R. (2023). Elon Musk's Shadow Rule. *The New Yorker*.

Fiscal Note Inc. (2025). *Donald Trump: Social Media Archive*. Roll Call The Source for News on Capitol Hill since 1955. https://rollcall.com/factbase-twitter/?platform=all&sort=date&sort_order=desc&page=1

Foster, J., Çetinoglu, Ö., Wagner, J., Le Roux, J., Hogan, S., Nivre, J., Hogan, D., & Van Genabith, J. (2011). # hardtoparse: POS Tagging and Parsing the Twitterverse. *AAAI 2011 Workshop on Analyzing Microtext*, 20–25.

Francis, W. N. (1964). A standard sample of present-day English for use with digital computers. In *Report to the U.S. Office of Education on Cooperative Research Project No. E-007*.

Gaizauskas, R., Burnard, L., Clough, P., & Piao, S. (2003). Using the XARA XML-Aware corpus query tool to investigate the METER Corpus. *In Proceedings of the Corpus Linguistics 2003 Conference*, 227–236.

Gimpel, K., Schneider, N., O'connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., & Smith, N. A. (2011). Part-of-speech tagging for twitter: Annotation, features, and experiments. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. 2011 Association for Computational Linguistics, Portland, Oregon.

Grundmann, R., & Krishnamurthy, R. (2010). *The discourse of climate change: A corpus-based approach. Critical Approaches to Discourse Analysis Across Disciplines 4*, 125–146.

Gudavalli, A., & JayaLakshmi, G. (2025). DYNAMIC WEB SCRAPING THROUGH SELENIUM WEBDRIVER IN PYTHON. *International Journal of Humanities Education*, *13*(1), 12–18.

Gui, T., Zhang, Q., Huang, H., Peng, M., & Huang, X.-J. (2017). Part-of-speech tagging for twitter with adversarial neural networks. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2411–2420.

Gul, N., Indrasing, R. K., AK Sathyan, N. M., & Bharat, M. D. (2025). WEB SCRAPER FOR DATA EXTRACTION AND THREAT INTELLIGENCE. *Mohite Dnyanesh, WEB SCRAPER FOR DATA EXTRACTION AND THREAT INTELLIGENCE (April 10, 2025)*.

Hardie, A. (2012). CQPweb—Combining power, flexibility and usability in a corpus analysis tool. *International Journal of Corpus Linguistics*, *17*(3), 380–409.

Hardie, A. (2023). *Simple Syntax*. CQPweb. https://corpora.linguistik.uni-erlangen.de/cqpweb/doc/cqpweb-simple-syntax-help.pdf

Heinrich, P. (2025). *Cwb-ccc* (Version 0.13.0) [Computer software]. https://github.com/ausgerech-net/cwb-ccc

Heinrich, P., & Evert, S. (2020). Operationalising the Hermeneutic Grouping Process in Corpus-assisted Discourse Studies. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 33–44.

Hoffmann, S., & Evert, S. (2005). BNCweb (CQP-edition): The marriage of two corpus tools. *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, *3*, 177–195.

Honnibal, M., Montani, I., Van Landeghem, S., & Boyd, A. (2020). *spaCy: Industrial-strength Natural Language Processing in Python*. https://doi.org/10.5281/zenodo.1212303

Jiang, H., Hua, Y., Beeferman, D., & Roy, D. (2022). Annotating the Tweebank corpus on named entity recognition and building NLP models for social media analysis. *Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022)*, 7199–7208.

Kempter, M. (2024, October 14). *Was bedeutet 'woke'? - Bedeutung und Herkunft*. Stuttgarter Zeitung. https://www.stuttgarter-zeitung.de/inhalt.was-bedeutet-woke-mhsd.e98ad6e7-a8b7-42e8-aae7-7bb0563e0a36.html

Khder, M. A. (2021). Web scraping or web crawling: State of art, techniques, approaches and application. *International Journal of Advances in Soft Computing & Its Applications*, *13*(3).

Kilgarriff, A. (2009). Simple maths for keywords. *Proceedings of the Corpus Linguistics Conference 2009*, *6*, 41–55.

Kilgarriff, A., Baisa, V., Buvsta, J., Jakubívcek, M., Kovàr, V., Michelfeit, J., Rychlÿ, P., & Suchomel, V. (2014). The sketch engine. *University of Toronto Press*, *1*, 7–36.

Kilgarriff, A., & Kosem, I. (2012). *Corpus tools for lexicographers*. na.

Kong, L., Schneider, N., Swayamdipta, S., Bhatia, A., Dyer, C., & Smith, N. A. (2014). A dependency parser for tweets. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1001–1012.

Kunilovskaya, M., & Plum, A. (2021). Text preprocessing and its implications in a digital humanities project. *Proceedings of the Student Research Workshop Associated with RANLP 2021*, 85–93.

Lehmann, H. M., & Schneider, G. (2000). *BNC DependencyBank 1.0. 12*.

Liu, Y., Zhu, Y., Che, W., Qin, B., Schneider, N., & Smith, N. A. (2018). Parsing tweets into universal dependencies. *arXiv:1804.08228*.

Lyndell, D. (2025). *Elon Musk Tweets 2010-2025 (April)*. Elon Musk Tweets 2010-2025 (April). https://www.kaggle.com/datasets/dadalyndell/elon-musk-tweets-2010-to-2025-march/data

Mahlberg, M., Stockwell, P., Joode, J. de, Smith, C., & O'Donnell, M. B. (2016). CLiC Dickens: Novel uses of concordances for the integration of corpus stylistics and cognitive poetics. *Corpora*, *11*(3), 433–463.

Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Association for Computational Linguistics 1993*, *273*, 31.

Mautner, G. (2009). *Corpora and critical discourse analysis* (Contemporary Corpus Linguistics). Paul Baker.

Melyawati, N. L. P., Asana, I. M. D. P., Putri, N. W. S., Atmaja, K. J., & Sudipa, I. G. I. (2024). Comparison of Automation Testing On Card Printer Project Using Playwright And Selenium Tools. *Journal of Computer Networks, Architecture and High Performance Computing*, *6*(3), 1309–1320.

Miklaszewicz, A. (2023). *Assessing Leadership in Business: A Critical Investigation of Elon Musk* (Honors Scholar Theses, p. 953). https://opencommons.uconn.edu/srhonors_theses/953

Nivre, J., De Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajic, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., & others. (2016). Universal dependencies v1: A multilingual treebank collection. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 1659–1666.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., & Smith, N. A. (2013). Improved part-of-speech tagging for online conversational text with word clusters. *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, 380–390.

Pérez, M. C. (2024). Russia and Ukraine through the Eyes of ParlaMint 4.0: A Collocational CADS Profile of Spanish and British Parliamentary Discourses. *Proceedings of the IV Workshop on Creating, Analysing, and Increasing Accessibility of Parliamentary Corpora (ParlaCLARIN)@ LREC-COLING 2024*, 84–93.

Persson, E. (2019). *Evaluating tools and techniques for web scraping*.

Piperski, A., Dietsch, L., Dykes, N., Evert, S., & Mahlberg, M. (2025). *FlexiConc demonstrator: A front-end web app for structured concordance analysis*. RC21 - Reading Concordances in the 21st Century, Friedrich-Alexander-Universität Erlangen-Nürnberg.

Polak, S. A. (2024). Donald Trump and truth social: Media platforms making exclusionary worlds. In *Diversity issues in the USA* (pp. 137–147). Transcript Verlag. https://hdl.handle.net/1887/4175588

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., & Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 101–108.

Rayson, P. (2009). Wmatrix: A web-based corpus processing environment. *Computing Department, Lancaster University*, *7*.

Remus, S., Hintz, G., Biemann, C., Meyer, C. M., Benikova, D., Eckle-Kohler, J., Mieskes, M., & Arnold, T. (2016). EmpiriST: AIPHES-robust tokenization and POS-tagging for different genres. *Proceedings of the 10th Web as Corpus Workshop*, 106–114.

Rooney, J. W. (Director). (2022). *Best Scraping Combo? Use These In Your Projects* [Video recording]. https://www.youtube.com/watch?v=HpRsfpPuUzE

Schmidt, J.-H. (2022, November 25). *Auf Twitter ließ er Medien nach seiner Pfeife tanzen*. Leibniz-Institut Für Medienforschung Hans-Bredow-Stiftung. https://leibniz-hbi.de/auf-twitter-liess-er-medien-nach-seiner-pfeife-tanzen/

Scott, M. (2024). WordSmith Tools version 9 (64 bit version). *Stroud: Lexical Analysis Software*.

Straka, M., & Straková, J. (2017). Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 88–99.

Stubbs, M. (1995). *Collocations and semantic profiles: On the cause of the trouble with quantitative studies*. *Functionsof Language 1*, 23–55.

Thole, A. N. (2022). *TIKTOK FORENSIC SCRAPER TO RETRIEVE USER VIDEO DETAILS* [PhD Thesis]. Purdue University Graduate School.

Wang, G., & Huan, C. (2023). *Negotiating climate change in public discourse: Insights from critical discourse studies*. *Critical Discourse Studies 21*, 133–145.

Wartena, C. (2023). *The hanover tagger (version 1.1. 0)-lemmatization, morphological analysis and POS tagging in python*.

Weisser, M. (2020). *Web as Corpus Resources* [Html]. Web as Corpus Resources. https://martin-weisser.org/corpora_site/web_corpus.html

Wodak, R. (2015). *The Politics of Fear: What Right-Wing Populist Discourses Mean*. SAGE, London.

Wysmolek, J. (2024). *Was ist ObamaCare?* https://signal-iduna.de/ratgeber/private-krankenversicherung-obamacare/#:~:text=Im%20Jahr%202010%20wurde%20in,in%20den%20USA%20lebenden%20Personen.

Yuan, S. & others. (2023). Design and visualization of Python web scraping based on third-party libraries and selenium tools. *Academic Journal of Computing & Information Science*, *6*(9), 25–31.

Zih, H., El Biadi, M., & Chatri, Z. (2022). Comparing LancsBox and AntConc in the extraction of Passives and Nominals: Towards Objectivity in Critical Discourse Analysis. *International Journal of Information Science and Technology*, *6*(2), 40–47.

7. Anhang

Folgendes sind alle Queries, mithilfe derer die verschiedenen Kommunikationsstrategien analysiert wurden:

1. hoax, conspir.*, illegal, legal
2. Joe, Biden, Hillary, Clinton, Harris, Kamala, Obama, Barack, China, Russia, Putin, loser, hater, IQ, corrupt, weak, idiot, fake
3. seem, might, doubt, real.*, wonder, maybe, perhaps
4. always, never, .*ly, incredibl.*
5. threat, crisis, attack, terror, danger, collapse, invasion, enemy, riot, will, steal
6. MAGA, eat
7. Pretend, lie, hypocri.*
8. America, Russian, woman, mexican, alien, democrat, republician, people, person, nation, patriot
9. Do, say, tell, claim, think, want, believe
10. Only, just, clear.*, simple, obvious
11. I, .*cause, Elon, engineer, scientist, expert
12. Fake news, deep state, media, woke
13. Good, bad, right, wrong, legal, illegal, light.*
14. .*like, as, .*esque
15. Siehe Keywords -> Notebook 04
16. Thing, all, each, situation, issue, problem, stuff
17. Meanwhile, different, besides, however, irrelevant
18. Everyone, together, should, join, support, movement, must, let's
19. Liar, corrupt, crooked, crazy, criminal, fraud, traitor, disgrace
20. Freedom, safety, security, justice, truth, value, peace, loyalty, god
21. Safe, secure, free, danger, war, fight, wealth, health, money
22. Great, poor, legal, illegal, hero, winner, beautiful, smart, brilliant, icon.*

Die **Analysistrees von FlexiConc** und ausführlichere Erklärungen zu den gewählten Kommunikationsstrategien befinden sich in den beiden Jupyter Notebooks 05 und 06.

Alle Daten, der Code, Requirements für verschiedene virtuelle Umgebungen, die Jupyter Notebooks 01-06 und alle sonstigen Informationen zur Bachelorarbeit befinden sich im Repository TrumMus.

Repository: https://github.com/vivien3mueller/TrumMus

Dies sind die verwendeten Notebooks:

- 01 Datenbeschaffung+Vorverarbeitung_Trump.ipynb
- 02 Tagging_Testkorpus.ipynb
- 03 Vorverarbeitung_Musk+Tagger.ipynb
- 04 CWB_cwb-ccc.ipynb
- 05 FlexiConc_Analyse.ipynb
- 06 FlexiConc_Kommunikationsstrategien.ipynb

Die Codezeilen wurden durch Fehleranalysen unter Zuhilfenahme von ChatGPT verbessert. Außerdem sind die letzten beiden Notebooks nicht vollständig ausgeführt worden, da Jupyter Notebook mit so vielen Queries und Knoten seine maximal Kapazität erreicht. Die Notebooks hängen auf meinem Rechner und lassen sich nicht weiter bearbeiten. Daher wurde die Analyse mit den bis dato erfolgten Ergebnissen beschrieben und alle weiteren Queries sind kein Teil des Kapitel 4 der Bachelorarbeit.

Ein Blogbeitrag zur Arbeit mit FlexiConc befindet sich ebenfalls im Repository und wird wahrscheinlich in Kürze auf https://www.dhss.phil.fau.eu/research/current-projects/reading-concordances-in-the-21st-century-rc21/rc21-blogs/ veröffentlicht.

8. Eigenständigkeitserklärung

Hiermit versichere ich, _____ (Name) _____ (Matrikelnummer), die vorgelegte Arbeit selbstständig und ohne unzulässige Hilfe Dritter sowie ohne die Hinzuziehung nicht offengelegter und insbesondere nicht zugelassener Hilfsmittel angefertigt zu haben. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen und wurde auch von keiner anderen Prüfungsbehörde bereits als Teil einer Prüfung angenommen.

Die Stellen der Arbeit, die anderen Quellen im Wortlaut oder dem Sinn nach entnommen wurden, sind durch Angaben der Herkunft kenntlich gemacht. Dies gilt auch für Zeichnungen, Skizzen, bildliche Darstellungen sowie für Quellen aus dem Internet.

Mir ist insbesondere bewusst, dass die Nutzung künstlicher Intelligenz verboten ist, sofern diese nicht ausdrücklich als Hilfsmittel von dem Prüfungsleiter bzw. der Prüfungsleiterin zugelassen wurde. Dies gilt insbesondere für Chatbots (insbesondere ChatGPT) bzw. allgemein solche Programme, die anstelle meiner Person die Aufgabenstellung der Prüfung bzw. Teile derselben bearbeiten könnten.

Des Weiteren ist mir bekannt, dass die gemeinsame Bearbeitung der Aufgabenstellung mit anderen Personen in einem Raum oder mithilfe sozialer Medien eine unzulässige Hilfe Dritter im o.g. Sinne darstellt, wenn nicht ausdrücklich Gruppenarbeit vorgesehen ist. Jeder Austausch mit anderen Personen mit Ausnahme von Prüfenden und Aufsichtführenden während der Prüfungszeit über Aufbau oder Inhalte der Prüfung oder Informationen (z.B. Quellen) ist unzulässig. Gleiches gilt für den Versuch der jeweiligen Handlung.

Verstöße gegen die o.g. Regeln sind als Täuschung bzw. Täuschungsversuch zu qualifizieren und führen zu einer Bewertung der Prüfung mit „nicht bestanden“.

_____ _____

Ort, Datum Eigenhändige Unterschrift