Jan Oliver Rüdiger

CorpusExplorer v2.0 – Visualisierung prozessorientiert gestalten

Abstract Der CorpusExplorer v2.0 ist eine frei verfügbare Software zur korpushermeneutischen Analyse und bietet über 45 unterschiedliche Analysen/Visualisierungen für eigenes Korpusmaterial an. Dieser Praxisbericht gibt Einblicke, zeigt Fallstricke auf und bietet Lösungen an, um die tägliche Visualisierungsarbeit zu erleichtern. Zunächst wird ein kurzer Einblick in die Ideen gegeben, die zur Entwicklung des CorpusExplorers[1] führten, einer korpuslinguistischen Software, die nicht nur vielfältige Forschungsansätze unterstützt, sondern auch mit einem Fokus auf die universitäre Lehre entwickelt wird. Der Mittelteil behandelt einen der vielen Fallstricke, die im Entwicklungsprozess auftraten: Effizienz-/Anpassungsprobleme – bzw.: Was passiert, wenn Visualisierungen an neue Begebenheiten angepasst werden müssen? Da diese Lösung Teil des CorpusExplorers v2.0 ist, wird abschließend darauf eingegangen, wie unterschiedliche Visualisierungen zu denselben Datensätzen sich auf die Rezeption/Interpretation von Daten auswirken.

1. Von der Idee zur Anwendung

Die eigentliche Idee zur Entwicklung des CorpusExplorers entstand während meiner Magisterarbeit 2013. Zunächst wurden Mitarbeiter/innen und Doktorand/innen des *Instituts für Germanistik* an der *Universität Kassel* befragt, ob/wie sie korpuslinguistisch arbeiten, welche Tools eingesetzt werden und wie aktuelle Lösungen in konkreten Projekten aussehen. Fast alle nutzten korpuslinguistische Tools, mal mehr, mal weniger intensiv. Überraschend war die Vielzahl an Softwaretools, die in der Forschung kursieren. Auf Basis dieser Befragung wurden mehrere Workflows entwickelt, um die folgende Frage beantworten zu können: *„Was muss erledigt werden, um vom einfachen Text zu einem visuellen Endergebnis zu gelangen?"* Die einzelnen Arbeitsschritte lassen sich durch verschiedene Softwaretools erledigen. Das Endergebnis ist ein sogenannter *„Toolchain"* – also eine

1 Der CorpusExplorer kann über http://corpusexplorer.de kostenfrei bezogen werden.

Erschienen in: Noah Bubenhofer u. Marc Kupietz (Hrsg.): Visualisierung sprachlicher Daten. Heidelberg: Heidelberg University Publishing, 2018
DOI: https://doi.org/10.17885/heiup.345.474

Kette von Softwaretools, die in einer bestimmten Reihenfolge genutzt werden, um den Workflow vollständig abzudecken. Ein Workflow-Beispiel: Rohtexte bereinigen, annotieren, Frequenzen auszählen und abschließend als Grafik darstellen. Der exemplarische *Toolchain* könnte wie folgt aussehen: Texte werden mit *JEdit* (per Hand) bereinigt, danach mit dem *TreeTagger* annotiert, Frequenzen werden mit *AntConc* ausgezählt; für die Umsetzung als Diagramm wird schließlich *Microsoft Excel* verwendet[2]. Dieser *Toolchain* ist nur einer von vielen denkbaren Möglichkeiten, manche mögen kürzer, einfacher, detaillierter oder komplexer sein. Für alle ergeben sich aber ähnliche Probleme:

— Für alle Programme müssen Lizenzen erworben werden, insofern diese nicht kostenfrei verfügbar sind. Der Kostenfaktor spielt besonders dann eine Rolle, wenn die Software z. B. in einer Seminargruppe eingesetzt werden soll. Der oben aufgezeigte Toolchain ist in der Regel mit geringen Kosten realisierbar, da die Programme JEdit, TreeTagger und AntConc kostenfrei sind und Excel auf vielen Rechnern vorinstalliert/vorlizenziert ist. Ganz anders gestaltet sich aber der Fall, wenn z. B. Produkte wie MAXQDA, SPSS oder Tableau zum Einsatz kommen sollen – hier können schnell mehrere Hundert/Tausend Euro an Lizenzkosten fällig werden. Das Ausweichen auf kostenfreie Open-Source Lösungen ist erstrebenswert, erfordert aber eine sachkundige Auswahl und kann zur Verstärkung der in den folgenden Punkten aufgelisteten Problemen führen.
— Die Programme müssen untereinander kompatibel sein. Die Ausgabe des einen Programms ist wiederum die Eingabe des anderen. Wie die Umfrage ergab, ist dies eine häufig wahrgenommene Fehler-/Problemquelle. Oft wird diese durch selbstentwickelte/selbsterdachte Lösungen beseitigt. Diese Lösungen sind jedoch fragil, da z. B. im Falle der Aktualisierung eines der beiden Programme die individuelle Anpassung neu konfiguriert werden muss. Daher kann es notwendig sein, dass man in einer Projektgruppe / während einer Projektphase – genau überlegt, welche Programmversionen zum Einsatz kommen, dies abspricht und ggf. ältere Installationspakete bereithält, um ggf. auf die ältere Programmversion zurückwechseln zu können, wenn das Update Probleme bereitet. Im Idealfall wird der Toolchain in regelmäßigen Abständen vollständig auf einem Rechner getestet und erst dann auf alle Rechner des Projektteams/der Seminargruppe übertragen.

2 Auf die einzelnen Tools wird im Folgenden nicht weiter eingegangen. Die Bezugsquellen lauten: JEdit: http://bit.ly/1TOp23W TreeTagger: http://bit.ly/1bsE7eE AntConc: http://bit.ly/1VrkiQY Microsoft Excel: http://bit.ly/1sXVQM4 MAXQDA: http://bit.ly/2qJ1bbT SPSS: https://ibm.co/2rK6hmJ Tableau: http://tabsoft.co/2rb1fCF .

— Die Anzahl der benötigten Softwareprogramme wird schnell zweistellig. Wie bei einer Kette so gilt auch für den Toolchain – das Endresultat ist nur so stark wie das schwächste Glied. Wird z. B. eine Software nicht mehr weiterentwickelt, muss ggf. der gesamte Toolchain umstrukturiert werden. Dieses Problem betrifft sowohl kommerzielle als auch Open-Source-Software. Kleinere/Kurzfristige Projekte können dieses Problem vernachlässigen, größere/mehrjährige Projekte sind gut beraten, alle Softwareprodukte im Toolchain auf Ausfallmöglichkeiten zu überprüfen (z. B. keine Open-Source Software zu nutzen, die bereits bei Projektbeginn seit mehr als einem Jahr nicht mehr aktiv entwickelt wird), abzusichern (z. B. eine Kopie des Quellcodes anzulegen) oder Alternativen bereitzuhalten.
— Durch die große Anzahl an Software wird es zudem schwerer, dieses Wissen in die Lehre zu übertragen. Für jede zusätzliche Software müssen extra Handreichungen für die Studentinnen und Studenten geschrieben werden. Alle Programme müssen auf den Rechnern in der gleichen Version vorliegen. Unterschiedliche Programmversionen führen gelegentlich zu ganz anderen Ergebnissen (siehe oben – Updateproblem). Dozentinnen und Dozenten müssen zudem für eine Vielzahl an Programmen Hilfestellungen leisten können.

Diese Problemfelder, gerade auch was den Einsatz in der universitären Lehre anbelangt, lassen sich z. B. auch bei Bubenhofer (2011), Dipper (2011) und Zinsmeister (2011) wiederfinden. Daher ist davon auszugehen, dass diese Probleme nicht spezifische Probleme der Umfrageteilnehmer/innen sind, sondern einen weit größeren Nutzerkreis betreffen. Darauf aufbauend wurde eine Lösung, der CorpusExplorer, entwickelt. Der CorpusExplorer nutzt ausschließlich freie Software (Freeware/Open Source) – Installation und Konfiguration erfolgen vollautomatisch. Die Softwareauswahl, auf die der CorpusExplorer v2.0 zurückgreift, erfolgt nach den oben angegebenen Kriterien, d. h. Nutzung von Open-Source-Software (die aktiv weiterentwickelt wird), Toolchain-Test, bevor die Software verteilt wird und es stehen mehrere alternative Möglichkeiten bereit (z. B. existiert noch eine ganze Reihe alternativer Tagger, wenn einer der Tagger nicht mehr weiterentwickelt werden sollte. Dies erlaubt es zusätzlich, auch auf individuelle Nutzerpräferenzen einzugehen). Damit sind die Lizenz-, Konfigurations- und Kompatibilitätsprobleme aus Nutzersicht weitestgehend gelöst und die Forderung Zinsmeisters (2011, 72), dass „[für] den Einsatz in der Lehre ‚Download and Run'-Ressourcen, bei denen die Ressource als nutzungsfähiges Gesamtpaket bezogen werden kann […] vorzuziehen" sind, ist erfüllt.

Der CorpusExplorer bietet einen denkbar einfachen Toolchain, da er alle automatisierbaren Aufgaben mittels einer intuitiven Programmoberfläche löst: Programm starten, Korpus importieren und über 45 Visualisierungen nutzen.

Aufbereitung, Bereinigung, Chunking, Annotation, Abtrennen von Metadaten uvm. werden im Importprozess vollautomatisch durchgeführt. Dadurch wird es möglich, ein weiteres Problem zu lösen, den Einsatz in der Lehre – ein Programm, eine Erklärung – Dozentinnen und Dozenten können sich wieder auf die Linguistik fokussieren.

2. Plädoyer für dreistufige Visualisierungsprozesse

Visualisierungen sind meist das Sahnehäubchen einer wissenschaftlichen Publikation. Sie schmücken ein Paper aus oder fassen die detaillierten Ergebnisse optisch klar und verständlich auf einem Poster zusammen. Manchmal sind sie auch selbst Gegenstand der Forschung, dann geht es um Fragen der Ästhetik oder Wahrnehmung. Diese Aspekte der Informationsvisualisierung werden aber im Folgenden bewusst ausgeklammert, zum einen, weil sie im Gesamtkontext des Tagungsbandes mehrfach diskutiert werden, und zum anderen, weil dies den Artikelumfang sprengen würde. Diese Diskussion soll daher um einen völlig neuen Aspekt erweitert werden: Die Effizienz des Visualisierungsprozesses. Konkret bedeutet dies, Abläufe zu schaffen, die die Visualisierung erleichtern, da sie prozessorientiert entweder unterschiedliche Daten gleichartig aufbereiten, um so identische Visualisierungen zu befüllen, oder für gleichförmige Daten eine Vielzahl an Visualisierungen bereitstellen.

Der einfachste denkbare Ablauf ist ein zweistufiger Prozess. Damit es im späteren Verlauf zu keiner Verwirrung kommt, benenne ich beide Prozessschritte gleich so, wie sie später benötigt werden. Die beiden Prozesskomponenten lauten: „*View*“ und „*Model*“[3]. Dabei stellt das Model die eigentlichen Daten bereit. Das Model kann z. B. eine einfache CSV-Tabelle sein, die die Daten in strukturierter Weise enthält, oder ein Korpus, das ausgewertet wird. Die *View* ist die Visualisierung. In dieser simplen Form greift die *View* direkt auf das *Model* zu. Dieser zweistufige Prozess ist zweifelsfrei der einfachste und schnellste Weg, Daten in eine visuelle Form zu pressen. Der wesentliche Nachteil dieser Lösung ist die geringe Nachhaltigkeit. Alle Abfragen, Aggregationen und Funktionen wie Sortieren, Filtern oder Gruppieren werden der einen oder anderen Seite zugeschlagen. Verändern sich die Daten im Model strukturell, muss die *View* zwingend angepasst werden. Umgekehrt verhält es sich ähnlich – *View* und *Model* sind direkt miteinander verwoben. Dadurch können beide nur mit hohem Aufwand gegeneinander ausgetauscht werden. Doch dies ist wichtig, wenn gleichartige Visualisierungen auf unterschiedliche Daten oder

3 Beide Begriffe rekurrieren auf die Disziplin, aus der sie stammen, die Softwaretechnik. Für eine Einführung sei insbesondere Eilebrecht und Starke (2013) empfohlen.

unterschiedliche Visualisierungen auf gleiche Daten angewendet werden sollen. Dieses Problem hatte die erste Version des CorpusExplorers. Daher erfolgt das Plädoyer für einen mehrstufigen Visualisierungsprozess nicht aus einer Laune heraus, sondern basiert auf der Erkenntnis, der eine schmerzliche Erfahrung zugrunde liegt.

Der wesentliche Änderungsschritt ist das Hinzufügen einer zusätzlichen Zwischenebene. Softwaretechnisch könnte man darüber streiten, ob diese als *„Controller"*, *„Presenter"* oder *„ViewModel"* usw. bezeichnet wird[4]. Essenziell muss man aber nur begreifen, dass eine zusätzliche Ebene einige erhebliche Vorteile mit sich bringt. Wie bereits ausgeführt, ist bei einem zweistufigen Prozess nicht immer klar, wo und wie die eigentlichen Abfragen zu realisieren sind. Erst Abfragen machen aus den Daten eine visualisierbare Menge. In einem dreistufigen Prozess lässt sich diese Frage jedoch wesentlich klarer beantworten. Das Model stellt ausschließlich die reinen Rohdaten bereit, die View sorgt ausschließlich für die Darstellung, alles andere wird in die Zwischenebene verlagert. Dadurch wird es möglich, aus unterschiedlichen *Models* Daten zu beziehen, diese zu verbinden und in neuer, aggregierter Form an die View weiterzureichen.

Bei der Entwicklung des CorpusExplorer fiel letztlich die Entscheidung auf das sogenannte *„MVVM"*-Entwurfsmuster[5] *(„Model, View, ViewModel").* Überraschenderweise konnte ich bisher keine konkreten Aussagen in der Literatur finden, die sich mit Effizienzgewinnen einer derartigen Umstrukturierung befassen. Daher habe ich versucht, eine Abschätzung auf Basis der Historie des CorpusExplorers zu entwickeln. Zum jetzigen Zeitpunkt verfügt der CorpusExplorer über exakt 47 Visualisierungen. Ein zweistufiger Prozess würde bedeuten, dass 47-mal alles neu implementiert werden müsste. Durch die Umstellung auf den dreistufigen Prozess benötigt der CorpusExplorer jedoch nur 29 Views, um denselben Funktionsumfang abzudecken. Daher konnte allein durch diese simple Umstrukturierung 39,3% des anfallenden Programmieraufwands vermieden werden.

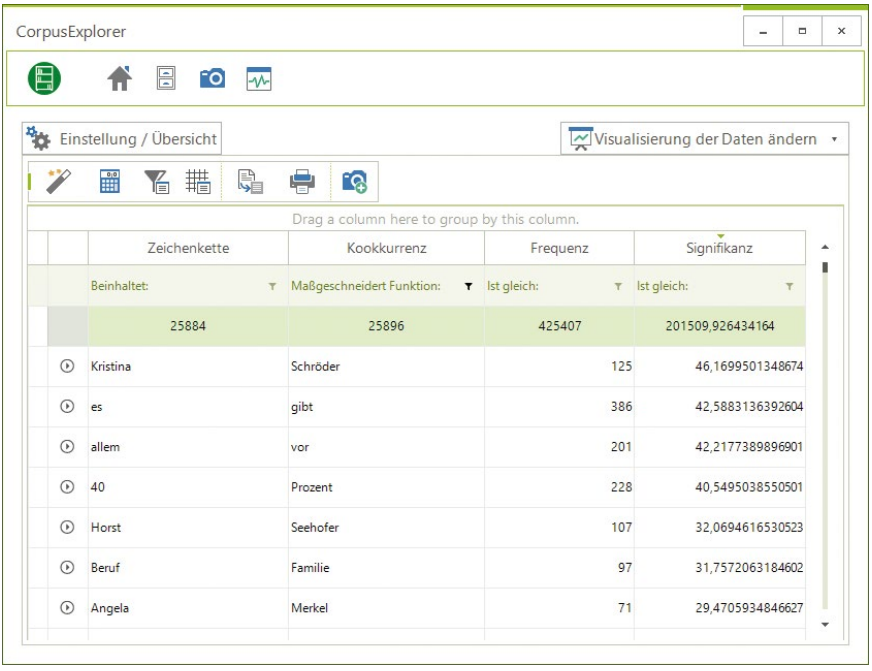
4 Konzeptionell gibt es zwischen diesen Zwischenschichtarten marginale Unterschiede. Diese hier zu diskutieren würde jedoch den Rahmen des Artikels sprengen.

5 Eine genau Definition des MVVM-Entwurfsmusters sowie eine Beispielimplementierung in C# findet sich bei Wegener und Schwichtenberg (2012, 583–609).

3. Identische Daten – Unterschiedliche Visualisierung

Kurz vorab: Für alle Beispiele wurde ein Korpus (1,28 Mio. Token) aus 2113 zufällig ausgewählten deutschsprachigen Zeitungsartikeln der Jahre 2010 bis 2015 zu den Stichworten Frauenquote/Quotenfrau verwendet. Alle Beispiele visualisieren das Resultat einer Kookkurrenz-Analyse. „Statistisch signifikante Kookkurrenzen sind Wortverbindungen, die überzufällig oft in einer bestimmten Datenbasis auftreten“ so Lemnitzer und Zinsmeister (2006, 147). Die erste und einfachste Form für die meisten Auswertungen ist die Darstellung in einer Tabelle. Im CorpusExplorer erlaubt die Tabellen-Ansicht das Sortieren, Filtern und Gruppieren der Daten. Abb. 1 zeigt die Tabellen-Ausgabe der Kookkurrenzanalyse.

Auch wenn Tabellen einen geringen visuell-ästhetischen Wert haben und keine Informationen raffen – im Vergleich zu anderen Visualisierungsformen haben sie jedoch einen eigenen Stellenwert im Visualisierungsprozess verdient. Tabellen sind der Überblick auf die Datengesamtheit. Durch Interaktivität können sie zudem sehr schnell zum gewünschten Analyseziel führen. Im Gegensatz zu anderen Programmen ermittelt der CorpusExplorer die Kookkurrenzen aller Token. Für die Signifikanz kann zwischen Poisson-Verteilung (Programmstandard), Chi-Quadrat-Test und Log-Likelihood gewählt werden. Nicht signifikante



Zeichenkette	Kookkurrenz	Frequenz	Signifikanz
Beinhaltet:	Maßgeschneidert Funktion:	Ist gleich:	Ist gleich:
25884	25896	425407	201509,926434164
Kristina	Schröder	125	46,1699501348674
es	gibt	386	42,5883136392604
allem	vor	201	42,2177389896901
40	Prozent	228	40,5495038550501
Horst	Seehofer	107	32,0694616530523
Beruf	Familie	97	31,7572063184602
Angela	Merkel	71	29,4705934846627

Abb. 1 Kookkurrenz-Tabelle im CorpusExplorer.

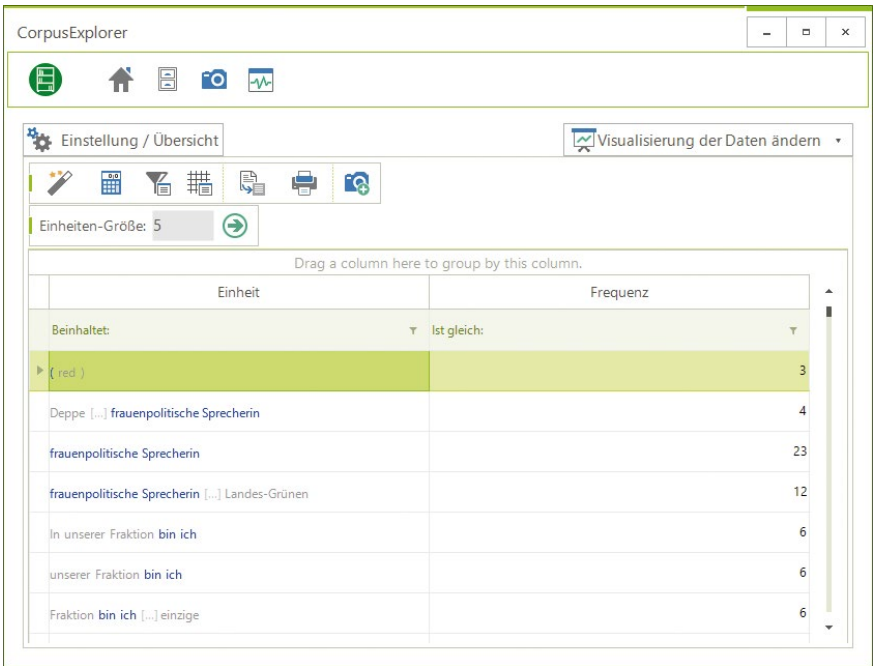


Abb. 2 N-Gramme mit Signifikanzwerten.

Kookkurrenzen werden automatisch gefiltert. Nutzerinnen und Nutzer können so entweder nach den signifikantesten Kookkurrenzpartnern suchen oder mithilfe der Filterfunktion Kookkurrenzen definierter Begriffe herausgreifen. Mit einer geringen Anpassung konnte aus dieser Auswertung eine weitere, weitaus differenziertere Darstellung entwickelt werden.

Abb. 2 zeigt eine weitere Tabelle. Die Idee entstammt dem Versuch, die Kookkurrenzanalyse aus COSMAS II[6] mit möglichst einfachen Mittel nachzubauen. Um diese Visualisierung zu realisieren, wurden zwei bereits vorhandene Auswertungen kombiniert. Hier spielt das *MVVM*-Konzept eine seiner größten Stärken aus, da bereits existierende Datenquellen leicht miteinander verknüpft werden können. Auf die Auswertung von N-Grammen[7] folgt eine Kookkurrenz-Analyse, diese bewertet, wie signifikant die Verbindungen einzelner Token innerhalb des

6 Online abrufbar über: http://www.ids-mannheim.de/cosmas2/

7 N-Gramme sind in der Zusammenfassung von Bubenhofer (2009, 118) wie folgt erklärt: „Das n steht für eine beliebige Zahl > 0; die Bezeichnung leitet sich von den Namen für Ein-, Zwei- oder Dreiwortausdrücke, ‚Unigramme', ‚Bigramme', ‚Trigramme', ab. Normalerweise werden n-Gramme nur als eine Reihe von direkt aufeinander folgenden Wörtern verstanden."

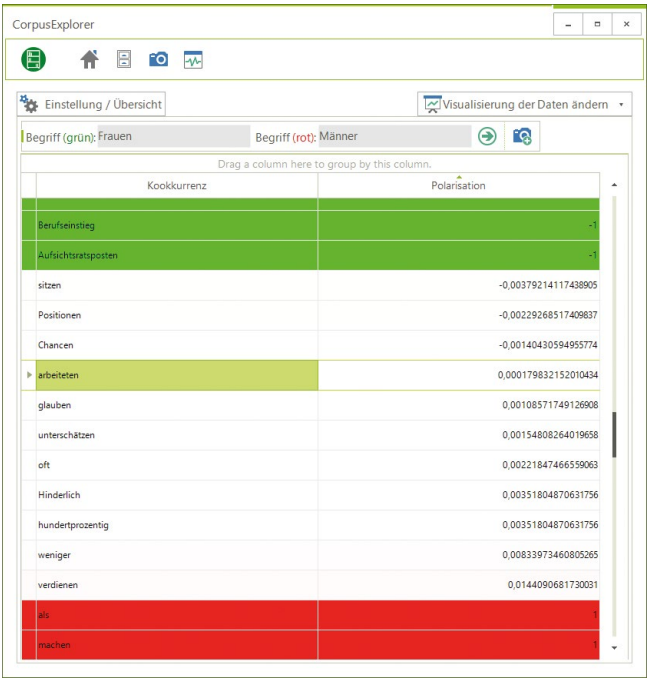


Abb. 3 Kontrastierte Kookkurrenzen. Grün = Frauen / Rot = Männer

N-Gramms sind. Je nach Signifikanzklasse erfolgt dann die Kolorierung. Blau steht für überdurchschnittlich signifikant, Schwarz für normal signifikant, Grau – unterer Grenzbereich der Signifikanz. Graue Auslassungszeichen stehen für nicht signifikante Token.

Abb. 3 zeigt die Kookkurrenzen zu lediglich zwei gewählten Begriffen, die gegeneinander kontrastiert werden.

Der erste Begriff (grün) lautet ‚Frauen', der zweite (rot) ‚Männer'. Diesmal werden jedoch anstelle der Gesamttabelle nur die zwei gewählten Begriffe in der View angezeigt. Dabei geht es um individuelle und überlappende Kookkurrenzen. Die Signifikanzwerte werden auf eine normierte Skala umgerechnet – die Normierung erfolgt im ViewModel. Alle Kookkurrenzen mit dem Wert gleich -1 gehören ausschließlich zum grünen Begriff – also ‚Frauen' – und sind folglich grün eingefärbt. Alle Begriffe mit dem Wert gleich +1 sind ebenso ausschließliche Kookkurrenzen zu ‚Männer'. Neben den ausschließlichen und damit individuellen Kookkurrenzen sind diejenigen besonders interessant, die zwischen diesen beiden Werten liegen – also im Wertbereich von -1 bis +1. Diese Begriffe sind signifikant zu beiden Begriffen. Durch das Vorzeichen ist die Tendenz zu einem der beiden Begriffe – man könnte auch sagen Pole – schnell identifizierbar.

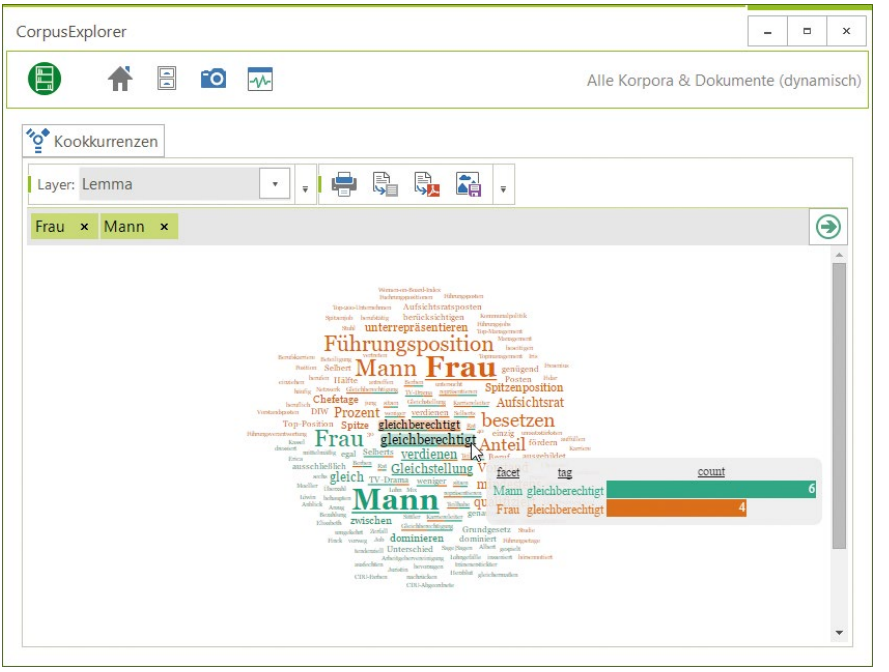


Abb. 4 Kookkurrenz-Wolke zu ‚Mann' und ‚Frau', visualisiert mit TagPies.

Jenseits von Tabellen bieten sich für Kookkurrenzanalysen natürlich noch weitere Darstellungsmöglichkeiten an. Abb. 4 zeigt die Visualisierung mittels TagPies, entwickelt von Jänicke et. al (2015), integriert über die HTML5-Schnittstelle des CorpusExplorers.

Die Datengrundlage ist absolut identisch mit der zuvor gezeigten Kookkurrenz-Kontrastierung. Die Möglichkeiten, die View/Visualisierung, sind aber um einiges umfangreicher, da diese auch mehrere ‚Pole' unterstützt, d. h. mehr als zwei Begriffe gegeneinander kontrastiert werden können. Sehr spannend ist an dieser Visualisierung auch die gegebene Interaktivität. Begriffe lassen sich anwählen und dadurch den einzelnen Polen zuordnen. Im Beispiel: ‚gleichberechtigt' tendiert eher zu ‚Mann' (6) als zu ‚Frau' (4).

Was wäre ein Artikel über Visualisierung in der Linguistik ohne Baumgraphen? – Abb. 5 zeigt einen solchen.

Die Abfrage ist identisch mit den vorherigen, Model und ViewModel konnten erneut eins zu eins wiederverwendet werden. Es erfolgt die Eingabe eines Begriffs, dann der Klick auf die Schaltfläche „Kookkurrenzen" und schon wird ein Baumgraph erstellt. Begriffe werden als farbige Knoten gezeichnet, die Kanten stellen die Relation (wer mit wem) her und geben Auskunft (siehe Werte)

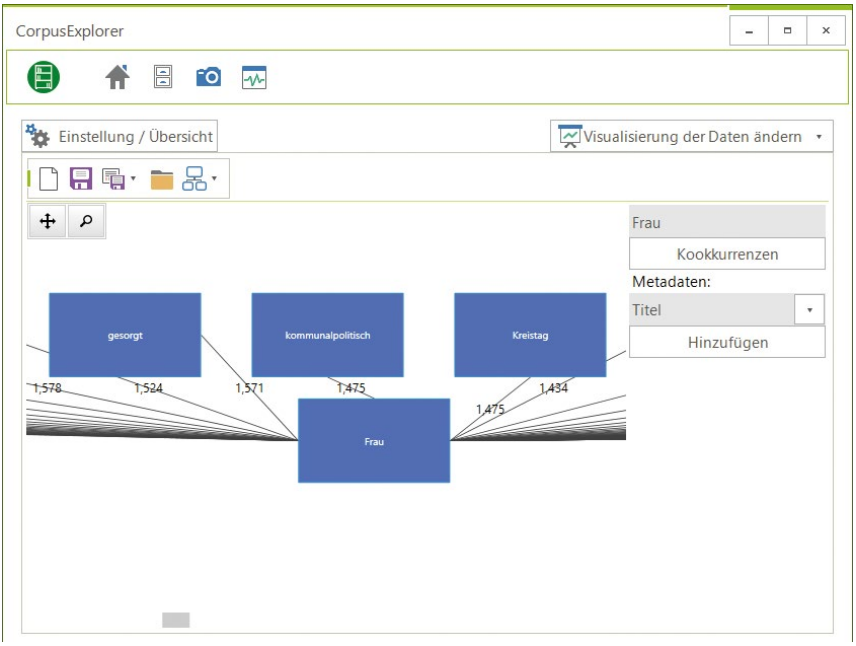


Abb. 5 Ausschnitt Kookkurrenz-Baum zu ‚Frau'.

über die Signifikanz der Verknüpfung. Würden weitere Begriffe wie z. B. ‚Kreistag' eingegeben, bekäme der Baum eine zusätzliche Tiefe – neue Begriffe werden mit existierenden verknüpft. Über die zusätzlichen Schaltflächen lässt sich der Graph layouten und einzelne Knoten/Kanten können zur besseren Darstellung entfernt werden (View-Funktionen). Zu den Begriffen lassen sich noch weitere Ressourcen einblenden wie etwa Metadaten zu den Dokumenten, so können z. B. Autoren oder Verlage als Knoten eingebunden und automatisch mit den Begriffen/Kookkurrenzen verknüpft werden. Auf diese Weise lässt sich in dieser Visualisierung mehr zeigen als ein bloßer Kookkurrenzgraph. Vielmehr erlaubt es Autoren-/Verlagstypiken auf der Mehrwortebene zu analysieren.

4. Fazit

Abschließend möchte ich zuerst festhalten: Der CorpusExplorer stellt keinesfalls eine *Ultima Ratio* für die Korpuslinguistik dar. Vielmehr geht es um einen kritikfähigen Beitrag, der einladen soll, über technische, theoretische und methodische Fragestellungen zu diskutieren. Wie gezeigt werden konnte, macht es Sinn, über Effizienz und Flexibilität im Visualisierungsprozess nachzudenken. Fast die

Hälfte an Ressourcen, hauptsächlich Zeit, konnte durch den dreistufigen Ansatz eingespart werden. Die gezeigten Visualisierungen machen außerdem recht deutlich, wie vielfältig sich die Ergebnisse ein und derselben Methode (Kookkurrenz-Analyse) darstellen lassen. Durch Verknüpfungen mit anderen Daten, auch innerhalb desselben Datensatzes – sowie durch Perspektivwechsel (zeige nur einige wenige, dafür aber spezifische Ergebnisse oder zeige das große Ganze) können Bedeutungsrelationen erzeugt werden, die die Datenrezeption beeinflussen. Aus meiner eigenen Seminarerfahrung heraus kann ich berichten, dass der Einsatz des CorpusExplorers nicht nur Freude in der Anwendung bereitet, sondern auch teilweise überraschende Ergebnisse in einem ansonsten unübersehbaren Berg aus Textmaterial zu Tage fördert. Anfangs herrscht Skepsis darüber, welchen Mehrwert solche Programme liefern können. Aus der Skepsis wird schnell Überraschung darüber, wie simpel doch manche Methoden sind und wie einfach es ist, diese Auswertungen selbst zu produzieren. Dies genügt meist, um Eifer zu entfachen hinter die Dinge sehen zu wollen – Belegstellen zu erkunden und Abfragen in immer komplexerem Maße zu kombinieren. Am Ende eines Seminars mit dem CorpusExplorer stehen motivierte Studierende, verwundert darüber, selbst empirische Forschung betrieben zu haben.

5. Bibliografie

Biemann, Christian. 2003. „Extraktion von semantischen Relationen aus natürlichsprachlichem Text mit Hilfe von maschinellem Lernen." LDV Forum – GLDV Journal for Computational Linguistics and Language Technology 18 (1/2): 12–25.

Bubenhofer, Noah. 2009: Sprachgebrauchsmuster. Berlin: de Gruyter 2009, 1:404. http://www.zora.uzh.ch/id/eprint/111287/1/BubenhoferSprachgebrauchsmusterPub.pdf.

Bubenhofer, Noah. 2011. „Korpuslinguistik in der linguistischen Lehre. Erfolg und Misserfolge." Journals for Language Technology and Computational Linguistics 26 (1): 141–156.

Dipper, Stefanie. 2011. „Digitale Korpora in der Lehre. Anwendungsbeispiele aus der Theoretischen Linguistik und der Computerlinguistik." Journals for Language Technology and Computational Linguistics 26 (1): 81–95.

Eilebrecht, Karl und Gernot Starke. 2013. Patterns kompakt: Entwurfsmuster für effektive Software-Entwicklung. 4. Aufl. Berlin: Springer Vieweg (IT kompakt). E-Book.

Heyer, Gerhard, Uwe Quasthoff und Thomas Wittig. 2006. Text Mining: Wissensrohstoff Text. Konzepte, Algorithmen, Ergebnisse. Herdecke: W3L-Verl. (IT lernen).

Jänicke, Stefan, Judith Blumenstein, Michaela Rücker, Dirk Zeckzer und Gerik Scheuermann. 2015. Visualizing the Results of Search Queries on Ancient Text Corpora with Tag Pies. http://docplayer.net/23646659-Visualizing-the-results-of-search-queries-on-ancient-text-corpora-with-tag-pies.html (letzter Zugriff am 01. Dezember 2017)

Lemnitzer, Lothar und Heike Zinsmeister. 2006. Korpuslinguistik eine Einführung. Tübingen: Narr (Narr-Studienbücher).

Moreaux, M. A. 1997. „Computerlinguistik für Philologen." LDV Forum – GLDV Journal for Computational Linguistics and Language Technology 14 (2): 22–27.

Runkler, T. A. 2010. Data Mining. Methoden und Algorithmen intelligenter Datenanalyse. Wiesbaden: Vieweg+Teubner.

Wegener, Jörg und Holger Schwichtenberg. 2012. WPF 4.5 und XAML: Grafische Benutzeroberflächen für Windows inkl. Entwicklung von Windows Store Apps. München: Hanser. E-Book.

Zinsmeister, Heike. 2011. „Chancen und Probleme der Nutzung von Korpora, Taggern und anderen Sprachressourcen in Seminaren." Journal for Language Technology and Computational Linguistics (JLCL) 26 (1): 67–79.