Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Visuelle Analyse gemeinsamer Sequenzen

Marcel Galuschka

Studiengang:	Informatik
Prüfer:	Prof. Dr. Thomas Ertl
Betreuer:	Moritz Knabben, David Schmid
Beginn am:	26. November 2019
Beendet am:	14. Juli 2020

Kurzfassung

In den sozialen Medien verbreiten sich Texte schnell. Dieses Phänomen lässt sich auch in historischen Zeitungsartikeln erkennen. Forscher aus der Literaturwissenschaft und der Kulturwissenschaft befassen sich mit diesem Thema und untersuchen eine Menge von Texten. In dieser Arbeit erstellen wir dazu einen Prototyp, mit welchem es möglich ist, diese Texte untereinander zu vergleichen und zwei Texte im Vergleich genauer zu betrachten. Außerdem diskutieren wir hier, aus welchem Grund wir diese Darstellung gewählt und wie wir diese verwirklicht haben. Darüber hinaus haben wir diesen Prototyp einer Literaturwissenschaftlerin vorgestellt, welche mit diesem sprachliche Veränderungen zwischen den Texten erkennen kann.

Inhaltsverzeichnis

Abbildungs- und Tabellenverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In historischen Zeitungen verbreiteten sich Artikel, indem sie in verschiedenen Zeitungen übernommen wurden. Forscher der Literaturwissenschaft sind an diesem Phänomen interessiert und analysieren diese Zeitungsartikel. Diese Menge an Artikeln lässt sich nur schwer von Menschen durcharbeiten. Dieses Problem möchten wir mithilfe einer Visualisierung lösen, indem wir die Ähnlichkeit von bis zu 90 Texten darstellen und analysierbar machen. In Zeitungsartikeln aus dem 19. Jahrhundert war das Zitieren von anderen Artikeln noch nicht verbreitet.

Dies führte dazu, dass Zeitungsartikel aus dieser Zeit aus anderen entstanden und Textabschnitte davon kopiert worden sind. Die Linguisten möchten nun jedoch herausfinden, wann und wo ein Autor Teile eines Artikels hinzugefügt oder verändert hat. Der Prototyp, welcher in dieser Arbeit erstellt wurde, soll dabei helfen, die Textstellen der Artikel ausfindig zu machen. Im Zuge dieser Arbeit entwickeln wir einen Prototyp zur Visualisierung von 50 bis 90 Zeitungsartikeln. Diese Visualisierung basiert darauf, dass die Artikel bereits gegenseitig viele ähnliche Textabschnitte beinhalten.

Aktuelle Visualisierungen von Textkopien, wie z.B. in der Arbeit *Visualizations for Text Re-use* [JGBS15] vorgestellt wurden, basieren auf dem Vergleich von einer kleinen Anzahl von maximal 30 Texten. Bei diesen werden zwei Texte in einer Matrix verglichen. Dabei unterscheidet der Prototyp zwischen einer Kopie, welche nur einmal geklont wurde und Textabschnitten, welche an mehreren Stellen im Text wieder aufgriffen wurden. Außerdem lassen sich einzelne, in sich abgeschlossene Abschnitte, welche sich nur in der Formulierung unterscheiden, in einem Graphen anzeigen. Der Fokus liegt dabei, aus den einzelnen Formulierungen einen Basistext zu erkennen. Ein hybrider Ansatz von einer Adjazenzmatrix und einem Node-Link-Diagramm soll, laut Henry, Fekete und McGuffin [HFM07], auch bei über 120 Artikeln helfen, den Überblick zu bewahren. Dieser vereint Cluster im Datensatz zu einer Adjazenzmatrix, verbirgt jedoch weniger stark verwandte Knoten mit einem einzelnen Link und spart dadurch Platz auf dem Bildschirm.

Da wir für unser Problem weder Datensätze mit wenigen Artikeln haben noch automatisch erkennen können, welche Texte stärker verwandt sind als andere, haben wir uns dazu entschieden, den gesamten Datensatz in einer Adjazenzmatrix zu visualisieren. In dieser Matrix vergleichen wir jeden Text mit allen anderen Texten. Um mehr Informationen auf der gleichen Fläche darstellen zu können, teilen wir die Adjazenzmatrix in zwei Teile auf. Diese beiden Teile unterscheiden sich farblich und sind durch eine Diagonale getrennt. Zusätzlich zu dieser globalen Ansicht lassen sich die Knoten und die Kanten der Adjazenzmatrix anklicken. Damit ist es möglich – zusätzlich zu dieser Matrix – zwei Texte direkt und mehrere Texte mit den übrigen Texten im Datensatz zu vergleichen. Der Vergleich von zwei Texten ist in Abbildung 1.1 dargestellt.

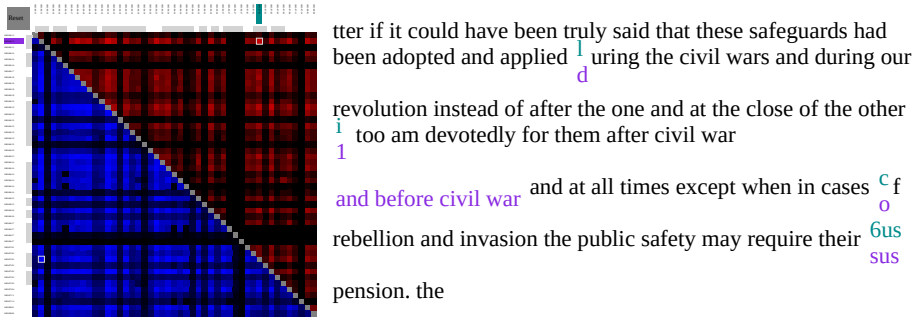


Abbildung 1.1: Auf der linken Seite ist der gesamte Datensatz abgebildet. Auf der rechten Seite ist in dieser Abbildung ein direkter Vergleich von zwei Texten angezeigt.

Für diese Visualisierung benötigen wir Metriken, mit welchen wir Texte untereinander auf gemeinsame Abschnitte testen können. Dafür verwenden wir die Longest Common Subsequence (LCS) sowie den Jaccard-Koeffizienten von Zeichen-N-Grammen. Außerdem nutzen wir für den Vergleich von Texten im Bezug auf den restlichen Datensatz die Metrik aus der Arbeit *Comparing Corpora using frequency profiling* [RG00], mit welcher es möglich ist, bezeichnende Wörter für Textkörper zu finden. Diese Metriken haben bei unseren Datensätzen eine Laufzeit von über einer Stunde. Aus diesem Grund berechnen wir diese im Vorfeld, um auch für große Datensätze mit bis zu 100 Artikeln eine schnellere Visualisierung gewährleisten zu können.

Auf diese Weise haben wir einen Prototyp entwickelt, welcher uns ermöglicht, verwandte Texte aus einem Datensatz zu erkennen. Es kann ein Text mit einem anderen verglichen werden. Außerdem ist es möglich, mehrere Texte mit den übrigen Texten aus diesem Datensatz zu vergleichen, wobei Worte angezeigt werden, die diese beiden Textgruppen unterscheiden.

2 Grundlagen

In diesem Kapitel beschreiben wir die Grundlagen dieser Arbeit. Diese bestehen aus den Grundkomponenten, die zur Visualisierung führen und Metriken, mit welchen wir die Texte analysieren. Bei unseren Daten suchen wir nach Abschnitten, die in den Artikeln hinzugefügt, geändert oder entfernt wurden. Dazu vergleichen wir immer zwei Texte miteinander, bis jeder Text mit jedem verglichen wurde. Um Ähnlichkeit von Textausschnitten darzustellen, verwenden wir hauptsächlich zwei Metriken, die LCS und den Jaccard-Koeffizienten. Zudem vergleichen wir auch noch Gruppen von Texten, die vom Nutzer definiert werden. Dazu nutzen wir die Metrik, welche im Abschnitt 2.4 beschrieben ist.

2.1 Adjazenzmatrix

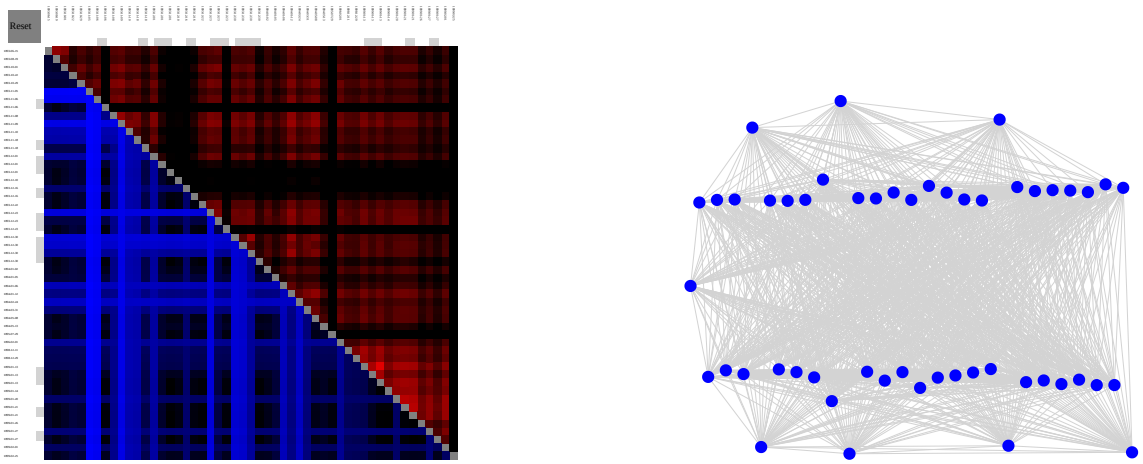
Die Adjazenzmatrix [TT13][S. 421] ist eine Möglichkeit, einen Graphen darzustellen. Dabei wird für jeden Knoten eine Spalte und eine Zeile in einer Matrix angelegt. Jeder Eintrag in der Matrix gehört zu je zwei Knoten und repräsentiert die Kante zwischen diesen Knoten. Wir vergleichen jeden Text mit jedem und erhalten so einen vollständigen Graphen, dabei repräsentiert jede Kante einen Vergleichswert zweier Texte, was wir in Kapitel 4 genauer beschreiben.

Neben der Adjazenzmatrix ist das Node-Link-Diagramm eine weitere Variante, wie ein Graph visualisiert werden kann. Bei diesem wird jeder Knoten auf einer zweidimensionalen Fläche verteilt, die Kanten zwischen den Knoten werden als Linie zwischen zwei Knoten angezeigt.

Bei unseren Daten handelt es sich um einen vollständigen Graphen. Demnach hat jeder Knoten eine Kante zu jedem anderen Knoten im Graphen. Da bei einem Graph mit mehr als vier Knoten die Kanten im Node-Link-Diagramm dann übereinander liegen müssen, lassen sich Kanten nicht auf den ersten Blick zu den Knoten zuordnen.

Für unser Problem geben uns die Knoten allein keine globale Information über den Datensatz. Wir würden so einen Artikel im Vergleich zu den übrigen Artikeln betrachten. Damit können wir einen Artikel, der nicht zum Rest der Artikel im Datensatz passt, finden. Dies ist jedoch nicht immer der Fall.

Das Verfolgen von einzelnen Kanten ermöglicht dem Nutzer zu erkennen, welche Texte untereinander Gemeinsamkeiten haben. Demnach vergleichen wir im Folgenden die Adjazenzmatrix sowie das Node-Link-Diagramm im Hinblick darauf, Kanten zu erkennen und bei gegebenen Kanten die dazugehörigen Knoten zu finden.



(a) Adjazenzmatrix (b) Node-Link-Diagramm

Abbildung 2.1: Vergleich Adjazenzmatrix und Node-Link-Diagramm. Hier sind 50 Knoten in einem vollständig verbunden Graphen dargestellt. Während in der Adjazenzmatrix die Kanten in rot und blau nachvollziehbar sind, sind im Node-Link-Diagramm Kanten nicht mehr nachverfolgbar.

In der Arbeit *Visualizing weighted networks: a performance comparison of adjacency matrices versus node-link diagrams* [MOB+12] wird darüber diskutiert, wann eine Adjazenzmatrix und wann eine Visualisierung mittels eines Node-Link-Diagramms sinnvoller ist. Hier wurden Studien durchgeführt, bei welchen diese beiden Arten der Visualisierung verglichen wurden. Aus diesen Studien lässt sich erkennen, dass eine Kante zwischen zwei Knoten mit einer Adjazenzmatrix im Schnitt deutlich schneller – in einem Drittel der Zeit – gefunden wird. Ebenso verhält es sich, wenn der Nutzer auf der Suche nach einem einzelnen Knoten ist. Hierbei ist jedoch zu beachten, dass aufgrund des geringen Abstands zu den anderen Knoten die Genauigkeit, den korrekten Knoten zu finden, in dieser Studie um ungefähr 10% sank. Beim Testen, welchen exakten Wert die farblich markierte Kante repräsentiert, konnten beide Visualisierungen gute Ergebnisse liefern. Dennoch ist in diesem Test die Genauigkeit bei der Adjazenzmatrix etwas geringer.

In unserem Fall nutzen wir die Adjazenzmatrix, da unser Graph vollständig verknüpft ist und sich somit in einem Node-Link-Diagramm sehr viele Kanten überschneiden würden. Damit wäre das Node-Link-Diagramm unübersichtlich. Dies wird bereits bei einem Datensatz mit 50 Knoten in Abbildung 2.1b sichtbar. Im Node-Link-Diagramm ist es nahezu unmöglich, eine Kante zu verfolgen. Der gleiche Datensatz ist in Abbildung 2.1a in einer Adjazenzmatrix dargestellt. In dieser Darstellungsform ist jede Kante der Matrix zu sehen. Außerdem sind in der Adjazenzmatrix die Knoten sortiert. Der Nutzer kann auch bei vielen Knoten noch nachvollziehen, wo die Kante zwischen zwei Knoten liegen muss.

2.2 Überdeckung der *Longest Common Subsequence*

Eine Möglichkeit, zwei Texte zu vergleichen, ist es, zu bestimmen, zu welchem Anteil der eine Text in dem anderen vorkommt. Dabei berechnen wir zunächst die LCS [Hir77]

Tabelle 2.1: Beispielrechnung der LCS von „acdagf“ und „cdgaf“. Die Tabelle wird zeilenweise ausgefüllt. Die Pfeile zeigen an, aus welcher Zelle die aktuelle Teilsequenz stammt.

	∅	a	c	d	a	g	f
∅	∅	∅	∅	∅	∅	∅	∅
c	∅	$\overset{\uparrow}{\leftarrow}_{\emptyset}$	$\nwarrow_{(c)}$	$\leftarrow_{(c)}$	$\leftarrow_{(c)}$	$\leftarrow_{(c)}$	$\leftarrow_{(c)}$
e	∅	$\overset{\uparrow}{\leftarrow}_{\emptyset}$	$\uparrow_{(c)}$	$\overset{\uparrow}{\leftarrow}_{(c)}$	$\overset{\uparrow}{\leftarrow}_{(c)}$	$\overset{\uparrow}{\leftarrow}_{(c)}$	$\overset{\uparrow}{\leftarrow}_{(c)}$
g	∅	$\overset{\uparrow}{\leftarrow}_{\emptyset}$	$\uparrow_{(c)}$	$\overset{\uparrow}{\leftarrow}_{(c)}$	$\overset{\uparrow}{\leftarrow}_{(c)}$	$\nwarrow_{(cg)}$	$\leftarrow_{(cg)}$
a	∅	$\nwarrow_{(a)}$	$\overset{\uparrow}{\leftarrow}_{(a)\&(c)}$	$\leftarrow_{(a)\&(c)}$	$\leftarrow_{(a)\&(c)}$	$\uparrow_{(cg)}$	$\overset{\uparrow}{\leftarrow}_{(cg)}$
f	∅	$\uparrow_{(a)}$	$\overset{\uparrow}{\leftarrow}_{(a)\&(c)}$	$\overset{\uparrow}{\leftarrow}_{(a)\&(c)}$	$\overset{\uparrow}{\leftarrow}_{(a)\&(c)}$	$\uparrow_{(cg)}$	$\nwarrow_{(cgf)}$

der beiden Texte. Diese ist die längste Sequenz an Zeichen, welche in beiden Texten in der gleichen Reihenfolge vorkommt. Betrachtet man zum Beispiel die Texte „acdagf“ und „cegaf“, ist die LCS die Sequenz „cgf“. Berechnet wird diese, indem die Tabelle 2.1 ausgefüllt wird. In dieser Tabelle steht ∅ für eine Sequenz ohne ein Zeichen. In Klammern sind jeweils die längsten Teilsequenzen aufgelistet und die Pfeile zeigen an, aus welcher Richtung die neuen Teilsequenzen stammen.

Die Länge dieser berechneten LCS teilen wir danach durch die Länge des kürzeren der beiden Texte. Ist $\mathrm{lcs}(t_1, t_2)$ die Funktion, welche die LCS der Texte t_1 und t_2 und $\min(t_1, t_2)$ die Länge des kürzeren der beiden Texte berechnet, so benutzen wir zur weiteren Analyse die folgende Formel

$$\mathrm{cover} = \frac{\mathrm{lcs}(t_1, t_2)}{\min(t_1, t_2)}.$$

Diese bildet die Überdeckung des kürzeren Textes von der Teilsequenz. In unserem Beispiel hätten wir eine Überdeckung von

$$\mathrm{cover} = \frac{\mathrm{lcs}(t_1, t_2)}{\min(t_1, t_2)} = \frac{\mathrm{lcs}(\mathrm{acdagf}, \mathrm{cegaf})}{\min(\mathrm{acdagf}, \mathrm{cegaf})} = \frac{3}{\min(5, 6)} = \frac{3}{5}.$$

Damit berechnen wir die Überdeckung der längsten Teilsequenz, was zu folgenden Eigenschaften führt: Ist der eine Text eine identische Kopie des anderen, in diesem Sinne also jedes Zeichen in dem einen Text auch an der gleichen Stelle wie im anderen, dann erhält diese Berechnung den Wert 1. Sollte ein Text eine Kopie des anderen sein, jedoch zusätzlich ein Absatz hinzugekommen sein, so wird die Berechnung ebenfalls eine vollständige Überdeckung und somit eine 1 zurückgeben.

Außerdem kann es vorkommen, dass beide Texte Abschnitte enthalten, die entweder nicht im anderen oder nicht an der gleichen Stelle im anderen vorkommen. In diesem Fall berechnet man, wenn l_{sec} die Länge des Abschnittes im kürzeren Text ist, mit

$$\mathrm{cover} = \frac{\min(t_1, t_2) - l_{\mathrm{sec}}}{\min(t_1, t_2)}$$

Tabelle 2.2: Mit dieser Tabelle beschreiben wir, wie wir die Zeichen-N-Gramme bilden. Im zweiten Text ist das „l“ fälschlicherweise als „i“ erkannt worden.

Text	Menge der Zeichen-N-Gramme (mit $n = 2$)	Menge der Wort-N-Gramme
sample text	{s, a, m, p, l, e, t, x, sa, am, mp, pl, le, te, ex, xt}	{sample, text, sample text}
sampie text	{s, a, m, p, i, e, t, x, sa, am, mp, pi, ie, te, ex, xt}	{sampie, text, sampie text}

die Überdeckung der Teilsequenz. Darüber hinaus können in einem Text beim Erkennen von Zeichen von gescannten Texten Fehler auftreten. Das Erkennen von diesen Zeichen nennt man Optical character recognition (OCR). Ist n_{error} die Anzahl der OCR-Fehler bei zwei identischen Texten, so ist die Überdeckung

$$\text{cover} = \frac{\min(t_1, t_2) - n_{\text{error}}}{\min(t_1, t_2)}.$$

Somit fließt ein Fehler linear zur Textgröße in die Skala von 0 bis 1 ein.

2.3 Jaccard-Koeffizient

Bei der Berechnung der Ähnlichkeit der Texte mit der längsten gemeinsamen Teilsequenz kommt es dazu, dass ein Hinzufügen von ganzen Absätzen zu einem Text nicht berücksichtigt wird. Aus diesem Grund nutzen wir noch eine andere Metrik, die dem Nutzer eine weitere Quelle gibt, die Zugehörigkeit zweier Texte abzuschätzen.

Der Jaccard-Koeffizient [Jac12] ist im Allgemeinen dafür gedacht, um die Überdeckung zweier Mengen zu bestimmen. Dabei berechnen wir den Jaccard-Koeffizienten j aus den beiden Mengen A und B wie folgt:

$$\text{j} = \frac{|A \cap B|}{|A \cup B|}$$

Da die Mächtigkeit der Vereinigung von zwei Mengen immer größer oder gleich der Mächtigkeit vom Schnitt dieser Mengen ist, ist auch der Jaccard-Koeffizient aus diesem Grund zwischen 0 und 1.

Da der Jaccard-Koeffizient mit Mengen arbeitet, müssen wir den Text in eine Menge aufteilen. Dabei haben wir uns für Zeichen-N-Gramme entschieden. Bei diesen teilen wir den Text in Teilstrings auf, die eine Länge von einem bis zu n Zeichen haben. In Tabelle 2.2 zeigen wir, wie wir die Zeichen-N-Gramme bilden. Im Gegensatz zu den Wort-N-Grammen teilen wir den Text nach n Zeichen und nicht nach n Wörten auf. Da wir den Text nicht auf Worte beschränken sondern einzelne Zeichen nutzen, haben wir ebenfalls Überlappungen bei den Mengen, auch wenn einzelne Zeichen beim Einscannen falsch erkannt wurden. Wir begrenzen die Teilstrings bewusst nach einigen Zeichen, hier nach 7 Zeichen, damit wir den Einfluss von einem einzelnen OCR-Fehler beschränken können. Ein OCR-Fehler führt im Nenner zu einem Wert, welcher um n größer ist als der Zähler. Dies würde für hohe n zu einem niedrigen Koeffizient führen, obwohl die Mengen identisch sein sollten. Mit

Tabelle 2.3: Variablenbenennung für die Berechnung Likelihood der Corpora Comparison.

Variable	Beschreibung
$w \in W$	Ein bestimmtes Wort aus dem gesamten Datensatz
nw_A	Anzahl des Wortes w in der ersten (markierten) Gruppe
nw_B	Anzahl des Wortes w in der zweiten Gruppe
$\mathrm{nw}_{\mathrm{total}}$	Anzahl des Wortes w in beiden Gruppen ($\mathrm{nw}_A + \mathrm{nw}_B$)
words_A	Anzahl aller Wörter in der ersten Gruppe
words_B	Anzahl aller Wörter in der zweiten Gruppe
$\mathrm{size}_{\mathrm{total}}$	Die Anzahl aller verschiedenen Wörter $\lvert W \rvert$

den beiden Texten aus der Tabelle 2.2 bekommen wir für die Zeichen-N-Gramme einen Jaccard-Koeffizienten j_z:

$$\mathrm{j}_z = \frac{13}{19}$$

Mit den Wort-N-Grammen bekommen wir einen Jaccard-Koeffizienten j_w:

$$\mathrm{j}_w = \frac{1}{5}$$

Da es sich hier um Mengen von Teilstrings handelt, wird ein Abschnitt, welcher in einem Text mehrfach vorkommt, nicht berücksichtigt. Kommt ein Textabschnitt jedoch nur in einem der beiden zu vergleichenden Texten vor, so wird dieser Abschnitt zur Berechnung des Koeffizienten berücksichtigt. Dies ist bei unserer Berechnung der LCS nicht der Fall.

2.4 Repräsentative Wörter von Textgruppen

Zusätzlich zum direkten Vergleich zwischen zwei Texten ist es auch möglich, mehrere Texte auszuwählen und diese im Vergleich zum Rest des Datensatzes zu sehen. Diese markierten Texte ergeben eine Gruppe von Texten, eine zweite Gruppe bilden die restlichen Texte des Datensatzes. Dazu möchten wir die Wörter finden, wodurch sich die beiden Gruppen an Texten unterscheiden. Hierzu stellen wir die Strategie vor, die von Rayson und Garside [RG00] erarbeitet wurde.

In den Texten können mehrere Texte markiert werden, die einen Textkörper bilden. Der Rest der Texte bildet den zweiten Textkörper. So ist es möglich, dass ein Textkörper deutlich kleiner ist als der andere. Daher ist es wichtig, dass der Algorithmus auch bei unterschiedlich langen Textkörpern stabil bleibt.

Ein Wort w repräsentiert eine der beiden Gruppen, wenn ein Wort in der Gruppe häufig vorkommt. Tritt dieses Wort jedoch auch im gesamten Datensatz oft auf, so wird dieses Wort als weniger relevant eingestuft. Genauer lässt sich dies an der Formel ablesen, wobei die Variablen in der Tabelle 2.3 nachgelesen werden können. Damit berechnen wir die Loglikelihood l für jedes Wort w im gesamten Datensatz mit folgender Formel:

$$l = 2 \cdot \left(\mathrm{nw}_A \cdot \log \left(\frac{\mathrm{nw}_A \cdot \mathrm{size}_{\mathrm{total}}}{\mathrm{words}_A \cdot \mathrm{nw}_{\mathrm{total}}} \right) \right) + \left(\mathrm{nw}_B \cdot \log \left(\frac{\mathrm{nw}_B \cdot \mathrm{size}_{\mathrm{total}}}{\mathrm{words}_B \cdot \mathrm{nw}_{\mathrm{total}}} \right) \right)$$

Ein Wort mit einer höheren Loglikelihood l ist wichtiger um einen der beiden Textkörper zu repräsentieren. In ihrer Arbeit haben Rayson und Garside [RG00] geschrieben, dass alle Wörter, die zu einem bestimmen Themengebiet gehören, markiert werden sollten. Da unsere Texte jedoch alle zu einem Themengebiet gehören, würde dies dazu führen, dass alle Wörter identisch markiert werden würden. Ebenso müssten diese Wörter manuell markiert werden, was für unsere Anwendung nicht nötig sein soll. So bekommen wir mit dem Algorithmus die Wörter heraus, die nur in der einen Textmenge vorhanden sind. Mit diesen Wörtern kann der Nutzer erahnen, welche Themengebiete in der einen und welche in der anderen Textgruppe stehen.

Zusätzlich zu diesem Algorithmus haben wir noch markiert, welches Wort welche der beiden Textmengen repräsentiert, da diese im Folgenden entsprechend markiert werden sollen.

3 Verwandte Arbeiten

In diesem Kapitel beschreiben wir, auf welchen Arbeiten unsere Visualiserung aufgebaut ist. Im Abschnitt 3.1 betrachten wir die Arbeit *Visualizations for Text Re-use* [JGBS15], in welcher die Analyse von einem zusammenhängenden Text beschrieben wird. In Abschnitt 3.2 präsentieren wir die Arbeit *The Diagram, a Method for Computing Sequences* [GM70], welche von der Darstellung von Sequenzen in Aminosäuren handelt. Außerdem betrachten wir eine Mischung von einer Adjazenzmatrix und einem Node-Link-Diagramm.

3.1 Visualizations for Text Re-use

Janicke et al. haben in ihrer Arbeit drei verschiedene Darstellungen für Text Re-use gezeigt [JGBS15]. Dort werden sieben unterschiedliche Übersetzungen der Bibel unter verschiedenen Aspekten von Textwiederholungen untersucht. Dabei handelt es sich im Vergleich zu unserem Problem mit bis zu 90 Artikeln um kleinere Datensätze.

3.1.1 Text Re-use Grid

Um eine Wiederverwendung von einzelnen Textabschnitten innerhalb eines Textes zu untersuchen, können beim *Text Re-use Grid* beispielsweise die ersten Bücher der Bibel analysiert werden. Bei dieser Visualisierung ist in einer Matrix zu sehen, welcher Teil des Textes wiederverwendet wurde. Zu jedem Teil des Textes wird die Anzahl, wie oft sich dieser Teil wiederholt sowie welche Art der Textwiederholung auftritt, angezeigt. Die Art unterscheidet dabei, ob viele Sätze des Textabschnittes wiederholt wurden oder nur ein paar wenige. In jedem Eintrag der Matrix sind demnach zwei Werte mithilfe einer Farbkodierung dargestellt. Die Sättigung der Zelle beschreibt, wie häufig der Abschnitt wiederholt wurde, die Farbe spiegelt die Art der Wiederholung wieder.

Hier werden zwei Texte miteinander verglichen und in einer Matrix dargestellt. In userem Fall möchten wir bis zu 90 Zeitungsartikel auf dem Bildschirm anzeigen. Dennoch verwenden wir einen ähnlichen Ansatz für unsere Adjazenzmatrix, wobei wir nicht jeden Textabschnitt, sondern jeden einzelnen Zeitungsartikel in dieser darstellen.

3.1.2 Text Re-use Browser

Mit dem *Dot Plot View* können einzelne Textwiederholungen einfacher typisiert werden. Dabei nennt man einen Text, der nur einmal vorkommt, eine systematische Kopie. Wird der Text hingegen an mehreren Stellen wiederverwendet, so nennt man dies wiederholende Textkopie. Jede Textwiederholung wird mit einem Punkt markiert. Anhand der Struktur der

Punkte lässt sich dann der Typ erkennen. Ist auf der Matrix eine diagonale Linie zu sehen, so ist der Text systematisch kopiert worden. Lässt sich auf der Matrix eine horizontale oder eine vertikale Anordnung von Punkten erkennen, dann wurde der Textabschnitt im Dokument öfter wiederholt.

Der *Text Re-use Browser* besteht ebenso aus einer Textanzeige. In dieser Darstellung werden beide Texte, die verglichen werden, nebeneinander angezeigt. Es wird für jede Textwiederholung, die gefunden wird, eine Linie eingezeichnet, die beide Abschnitte verbindet. Auch in dieser Ansicht lässt sich der Typ der Wiederholung erkennen. Sind viele parallele Linien zwischen den beiden Texten zu sehen, so handelt es sich um eine systematische Wiederholung. Führen zu einer Textstelle mehrere Linien, so ist in dem anderen Text der gleiche Abschnitt öfter vorhanden, der Abschnitt wurde also öfter wiederverwendet.

In unserem Fall suchen wir nicht nach einzelnen Textwiederholungen, sondern sind auf der Suche nach Stellen im Text, an denen die Texte nicht übernommen wurden. Bevor unsere Daten eingelesen werden, – welche aus einer Sammlung von historischen Zeitungsartikeln bestehen – wurde bereits eine Reprint-Analyse durchgeführt. Die Reprint-Analyse hat ergeben, dass in dem Datensatz sehr viele systematische Textwiederholungen vorhanden sind. Beim Anwenden dieser Visualisierung würden die vielen parallelen Verbindungen zwischen den Texten einer Analyse hinderlich sein.

3.1.3 Sentence Alignement Flow

Von unterschiedlichen Bibelversionen können Textabschnitte mit dem *Sentence Alignement Flow* [JGBS15] angezeigt werden. Hier werden die Textabschnitte so übereinander gelegt, dass die Wörter, die in mehreren Sätzen vorkommen, auch als ein Knoten in einem Graphen zu sehen sind. Verwendet ein Satz andere Wörter, so wird der Graph an diesen Wörtern aufgeteilt und jedes Wort wird übereinander angezeigt. So lässt sich der gesamte Abschnitt als Ganzes lesen, es lässt sich aber auch erkennen, welche Version bestimmte Satzteile anders formuliert hat.

Wir haben in unserem Fall um die 50 bis 90 Artikel, die wir analysieren möchten. Ein *Sentence Alignement Flow* auf 90 verschiedenen Artikeln ist zum einen in der Komplexität der Berechnung nicht vertretbar, zum anderen ist es auf handelsüblichen Bildschirmen nicht möglich, 90 Zeilen mit entsprechenden Abständen anzuzeigen. Wir verfolgen diesen Ansatz dennoch weiter, wenn wir zwei Texte aus dem Datensatz auswählen.

3.2 The Diagram

Zur Visualisierung von Sequenzen in Aminosäuren entwickelte Gibbs und Mcintyre *The Diagram* [GM70].

Bei dieser Visualisierung steht die Anzeige des Vergleiches von zwei Proteinsequenzen im Vordergrund. Diese werden in einer Matrix angezeigt, wobei ein Punkt in diesem Diagramm gezeichnet wird, wenn in beiden Sequenzen dasselbe Protein vorkommt.

Zusätzlich zu diesem direkten Vergleich zweier Sequenzen ist es noch möglich, mehrere Sequenzen untereinander zu vergleichen. Dies geschieht mit einem Dendogramm [Phi71]. In diesem Dendogramm sind die ähnlichsten Sequenzen jeweils Geschwister, was bedeutet, dass diese mit einer Linie verbunden werden. Die Gruppe, die die beiden bilden, kann dann wieder als neuer Kindknoten im Baum angesehen werden. Diese Ähnlichkeit wird darüber bestimmt, wie viele Punkte auf der Diagonalen im Diagramm liegen.

Für die Darstellung der einzelnen Sequenzen haben wir uns gegen eine Matrix in dieser Form entschieden. Diese Art der Darstellung skaliert schon für Texte mit mehr als 100 Zeichen nicht gut, da wir dafür 100 Zeilen auf dem Bildschirm benötigen würden. In Texten kommen viele verschiedene Zeichen vor, dies führt zu sehr dünn besetzten Matrizen. Aus diesen Matrizen können auf dem begrenzten Platz keine Aussagen gemacht werden. Außerdem handelt es sich bei dieser Matrix um einen Vergleich von zwei Proteinen. In unserem Fall bietet der Inhalt der Textsequenz weitere Informationen. Dies ist ein weiterer Grund dafür, dass wir es vorziehen, die beiden Texte als Text und nicht als weitere Matrix darzustellen.

Ebenso haben wir uns gegen die Variante entschieden, die Knoten in einem Dendogramm zu visualisieren. Dazu wäre es notwendig, automatisiert die Verwandtschaft von einzelnen Texten zu bestimmen. Diese Verwandtschaft ist bei unseren Texten jedoch generell sehr hoch, da die Texte bereits sehr ähnlich sind. Eine solche Klassifizierung zu automatisieren ist abhängig von dem Datensatz und nicht Teil der Arbeit.

3.3 NodeTrix

Eine Kombination eines Node-Link-Diagramms und einer Adjazenzmatrix wird in der Arbeit *NodeTrix a Hybrid Visualization of Social Networks* [HFM07] von Henry, Fekete und McGuffin vorgestellt.

Die Visualisierung basiert dabei auf einem Node-Link-Diagramm, wobei die Knoten dieses Diagramms wiederum eine Adjazenzmatrix beinhalten können. In der Arbeit wird ein Netz von Autoren aufgebaut, die zusammen an einer Arbeit geschrieben haben. Bei Autoren, welche oft zusammen auftreten, wird eine Gemeinschaft in Form einer Adjazenzmatrix aufgebaut. Die restlichen Verbindungen zwischen den Autoren werden in einem Node-Link-Diagramm visualisiert. Die Visualisierung der Gemeinschaften als Adjazenzmatrizen führt dazu, dass die stark vernetzten Teilgraphen mit möglichst wenigen Überlappungen dargestellt werden können. Zudem lässt sich der sonst nur wenig vernetzte Graph immer noch übersichtlich anzeigen. Eine vollständige Adjazenzmatrix bringt viele leere Zellen in der Matrix und wird damit unübersichtlich.

Wir würden diesen Ansatz gerne verfolgen, jedoch haben wir einen vollständigen Graphen. Damit müssten wir in den Artikeln engere Cluster finden, welche nur selten auftreten. Große nicht-triviale Cluster zu finden ist für einen Algorithmus nicht möglich. Dazu benötigen wir annotierte Dokumente, welche für unsere Einsätze nicht vorgesehen sind.

3.4 Word Cloud

Im Laufe der Arbeit möchten wir eine Menge an Wörtern anzeigen, welche zwei Sammlungen von Texten unterscheiden. Dazu haben wir uns die Word Cloud angesehen. Zu dieser Word Cloud haben Heimerl et al. [HLLE14] mit dem *Word Cloud Explorer* ein Tool erstellt, in welchem unter anderem verschiedene Layouts von Word Cloud angezeigt werden können.

Dabei ist zum einen die Ansicht verfügbar, bei welcher alle Wörter von der Mitte aus in einem Kreis angeordnet werden. Zum anderen ist es ebenso möglich, die Wörter in der Cloud nach dem Alphabet oder ihrer Gewichtung sortiert in Zeilen anzuzeigen. Das Layout, in welchem die Wörter nach der Gewichtung sortiert in Zeilen angezeigt werden, ist effizienter zu berechnen. Zusätzlich ist es für Daten, bei welchen die Gewichtung des einzelnen Wortes nicht relevant ist, sondern nur eine Tendenz dargestellt werden soll, auch möglich, die Schriftgröße für alle Wörter identisch zu lassen. Diese lassen sich dann auch platzsparend auf einer Fläche platzieren.

4 Visualisierungskonzept

In dieser Arbeit wollen wir eine Menge von ähnlichen Texten analysieren. Dabei untersuchen wir die Texte darauf, wann welche Textabschnitte geändert wurden. Mit unserem Konzept soll der Nutzer einen Überblick über die Ähnlichkeit der Texte im gesamten Datensatz gewinnen. Weiterhin soll es ihm möglich sein, zu erkennen, welche Texte eine höhere Überdeckung haben als andere, um damit detailliertere Informationen zu erhalten. Aus diesem Grund besteht unsere Benutzeroberfläche aus zwei Teilen. Auf der linken Seite steht eine quadratische Matrix, welche statisch an diesem Ort bleibt, die rechte Seite zeigt hingegen detailliertere Informationen über Teile der Daten, welche in der Matrix ausgewählt werden können.

Wir werden im weiteren das Konzept erläutern. Dazu haben wir einen Beispieldatensatz entwickelt. Dieser beinhaltet vier Sätze, die in der Tabelle 4.1 aufgelistet sind. Dabei sind die ersten beiden Sätze direkte Kopien und unterscheiden sich nur in einem Zeichen. Der dritte Text hat offensichtlich nichts mit den ersten beiden Texten zu tun. Der vierte Text ist eine Konkatenation des ersten und des dritten Textes.

4.1 Adjazenzmatrix

Damit der Nutzer effizient arbeiten kann, soll er möglichst wenige Texte lesen müssen. Aus diesem Grund erstellen wir eine Adjazenzmatrix, welche den gesamten Datensatz abbildet. Mit dieser Adjazenzmatrix, welche in Abbildung 4.1 zu sehen ist, soll es dem Nutzer möglich sein, einen Überlick über den Datensatz zu erhalten, ohne einen der Texte lesen zu müssen.

Hier sind die Artikel außerhalb der Matrix an dieser angrenzend nach ihrem Erscheinungsdatum geordnet. Sind zwei Artikel am selben Tag erschienen, so bekommen diese eine

Tabelle 4.1: Beispieldatensatz, den wir erstellt haben, um das Visualisierungskonzept erläutern zu können. Es handelt sich um 4 Texte, wovon zwei sehr ähnlich sind und einer sich davon deutlich absetzt.

Index	Datum	Text
1	02.06.1863	this is text 1 it is not that big
2	02.06.1863	this is text I it is not that big
3	03.06.1863	something completly different
4	04.06.1863	this is text 1 it is not that big something completly different

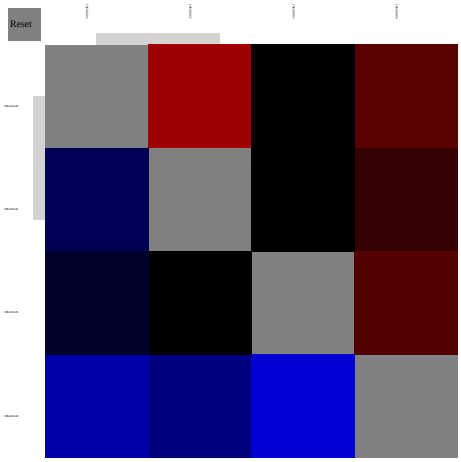


Abbildung 4.1: In dieser Abbildung ist unsere Adjazenzmatrix mit unserem Beispieldatensatz zu sehen. Oberhalb der Diagonalen sind die Jaccard-Koeffizienten und unterhalb der Diagonalen die Überdeckung der LCS abschätzbar.

Markierung am Rand der Matrix. Diese Markierung ist grau und verbindet alle Artikel die am selben Tag veröffentlicht wurden.

Bei der Analyse der Textartikel vergleichen wir paarweise alle Texte miteinander. Bei n Artikeln führt dies zu einem Aufwand in $\mathcal{O}(n^2)$, um die Vergleiche zu berechnen. Damit wir alle diese Vergleiche anzeigen können, steigt der Platz, welchen wir benötigen, ebenso mit n^2.

In der Adjazenzmatrix steht jede Spalte und jede Zeile für einen Text, welcher mit jedem anderen Text verglichen wurde. Jede Zelle beschreibt einen Textvergleich zwischen dem Text in welcher Zeile und dem Text in welcher Spalte sich diese Zelle befindet. Ein Block von angrenzenden Zellen mit hoher oder niedriger Helligkeit ist eine Menge von Texten mit besonders hoher beziehungsweise niedriger Ähnlichkeit. Betrachtet man eine Zeile im Verlauf über die Spalten, kann man einen Text verfolgen, wie dieser sich über die Zeit verändert hat. Das gleiche gilt für jede Spalte.

Oberhalb der Diagonalen ist in rot der Jaccard-Koeffizient abzulesen, dieser wird berechnet wie in Abschnitt 2.3 beschrieben. Im Beispiel in Abbildung 4.1 ist zu erkennen, dass die ersten beiden Texte einen hohen Jaccard-Koeffizient haben, da sie sich nur in einem Zeichen unterscheiden. Des weiteren kann man erkennen, dass weder der erste noch der zweite Text bezüglich der Jaccard-Metrik zum dritten Text gehören. Dass der dritte Text gerade so gewählt wurde, dass er nicht zu den ersten beiden passt, ist beabsichtigt. Zum vierten Text, welcher eine Konkatenation der beiden Textstämme ist, sehen wir in jeder Zelle der Matrix eine Rotfärbung, jedoch deutlich schwächer als zwischen den ersten beiden Texten. Auffällig ist hierbei, dass der Vergleich des ersten mit dem vierten im Gegensatz zum zweiten mit dem vierten eine höhere Übereinstimmung hat. Da wir in unserem Fall für diese Matrix keinerlei Sprachverarbeitung anwenden, können wir den zweiten Text als identischen Scan des ersten ansehen, bei welchem jedoch ein Zeichen einen OCR-Fehler enthält.

Unterhalb der Diagonalen ist in blau die Ordnung der längsten Teilsequenz kodiert. Hierbei nehmen wir die Überdeckung der längsten Teilsequenz zur kürzeren der beiden Texte.

Anhand dieser Überdeckung ordnen wir die einzelnen Vergleiche ein. Da in den Datensätzen aus einer Sammlung von Zeitungsreprints, die wir nutzen, im Allgemeinen sehr lange Teilsequenzen vorhanden sind, würde eine direkte Zuordnung von der Überdeckung in einen Farbwert zu vielen ähnlichen Werten führen. So wäre eine Aussage schwer zu treffen. In Abbildung 4.1 lässt sich ablesen, dass der erste sowie der zweite Text mit dem dritten Text keine lange gemeinsame Teilsequenz haben. Ebenso kann man erkennen, dass der dritte und der vierte Text eine lange gemeinsame Teilsequenz haben. Da der dritte Text gänzlich im vierten Text steht, ist 100% des Textes eine Teilsequenz. Genauso verhält es sich mit dem ersten Text im Vergleich zum vierten, jedoch wird dieser niedriger eingestuft, da wir eine strenge Totalordnung nutzen. Im Vergleich des zweiten Textes mit dem dritten ist die gemeinsame Teilsequenz am kleinsten, darum ist dieses Feld vollkommen schwarz.

Wir haben uns für eine farbige Darstellung entschieden, auch wenn die Information in Graustufen kodiert werden könnte. Dadurch, dass wir Farben verwenden, lässt sich der Wert einer Kante zum einen von der Diagonale unterscheiden, zum anderen ist in jedem Fall gegeben, dass sich der Matrixeintrag vom Hintergrund und von anderen Markierungen abhebt, da diese weiß und grau sind und diese Farben nicht in der Matrix vorkommen.

4.2 Textvergleich

Damit der Nutzer auch den Unterschied zwischen zwei Texten genauer betrachten kann, ist der Inhalt auf der rechten Seite der Visualisierung nicht statisch. In Abbildung 4.3 ist der direkte Textvergleich vom ersten und vom zweiten zu sehen. Die Beispieltexte sind in Tabelle 4.1 abzulesen. Ist der Textabschnitt in beiden Texten identisch, so ist er schwarz und mittig in der Zeile. Der Text teilt sich jedoch auf, sollte sich ein Teil in den beiden Texten unterscheiden. Diese Visualisierung ist direkt aus dem *Sentence Alignement Flow* inspiriert, der in Abschnitt 3.1.3 näher beschrieben ist.

Zu sehen ist in Abbildung 4.2 auf der linken Seite der Matrix in Violett markiert der Knoten, der auf der horizontalen Ebene liegt. Auf der vertikalen Ebene ist der Knoten über der Matrix in Cyan markiert. In derselben Farbe ist der Text dargestellt, welcher zu dem entsprechenden Text gehört. Um den markierten Matrixeintrag sowie um den korrespondierenden Eintrag auf der anderen Seite der Diagonalen ist eine hellgraue Umrandung. Damit soll es dem Nutzer möglich sein, den Jaccard-Koeffizient sowie die LCS auf einen Blick einzuordnen.

Zwischen zwei Texten kann es zu drei verschiedenen Szenarien kommen. Es können erstens in einem Text einige Zeichen oder Textabschnitte anders gewählt worden sein. Das lässt sich in Abbildung 4.3a erkennen. Hier spaltet sich der schwarz gedruckte Text in zwei Zeilen auf, oben steht der Abschnitt der im zweiten Text steht, darunter der Text der an der gleichen Stelle im ersten Text steht. Es kann zweitens ein ganz anderer Text stehen, wie in Abbildung 4.3b zu sehen. Der Text überschneidet sich zwar in einigen Zeichen, was aber nur Zufall ist und in Sprachen vorkommt. Außerdem können drittens Textabschnitte eingefügt werden wie in Abbildung 4.3c und in Abbildung 4.3d abgebildet. Hier ist ein Textabschnitt unten in Violett vorhanden. Der Teil, der sich in beiden Texten unterscheidet, kann also nur in einem der beiden Texte vorhanden sein. In einem anderen Fall könnte natürlich auch kein Text oberhalb zu sehen sein. Der Teil, der eingefügt wird, ist dann in

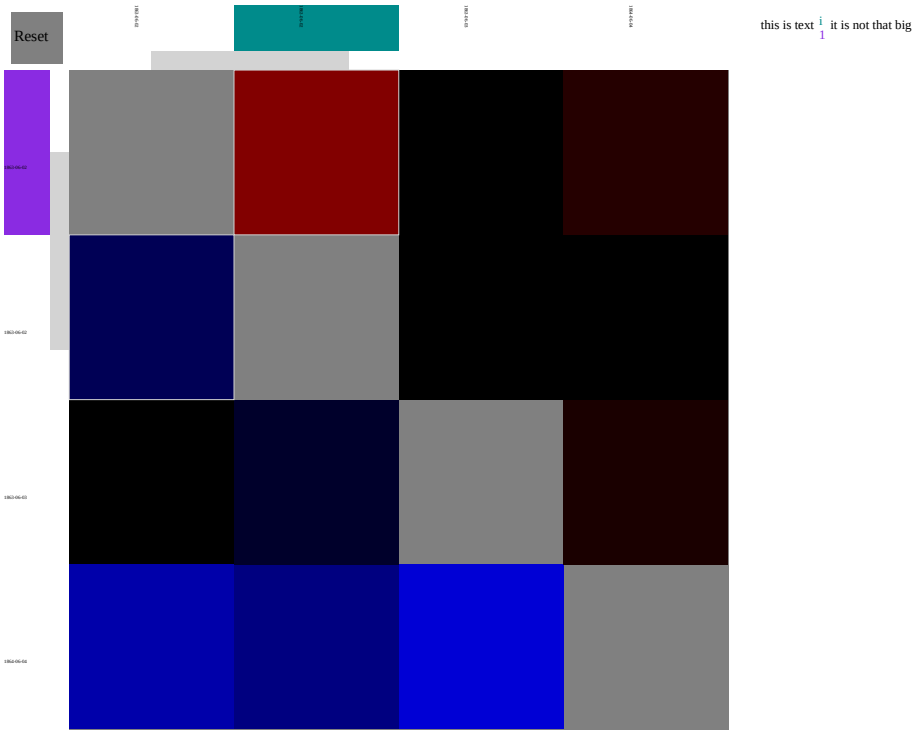


Abbildung 4.2: Abbildung eines Textvergleiches, bei welchem sich die Texte in einem Zeichen unterscheiden. Dies kommt bei einem OCR-Scan vor.

this is text i 1 it is not that big

(a) Textvergleich zwischen dem ersten und dem zweiten Text. Hier ist zu erkennen, dass sich die beiden Texte in einem Zeichen unterscheiden. Der Text teilt sich an dieser Stelle in zwei Zeilen auf.

some thi ng completly different
s is text 1 it is not that big

(b) Textvergleich zwischen dem ersten und dem dritten Text. Hier sind die zwei Zeilen nur in drei Zeichen vereint. Der Rest ist in beiden Texten unterschiedlich.

this is text 1 it is not that big something completly different

(c) Textvergleich zwischen dem ersten und dem vierten Text. Im ersten Teil unterscheiden sich die Texte nicht. Der zweite Teil in Cyan ist im vierten Text hinzugekommen.

this is text 1 it is not that big something completly different

(d) Textvergleich zwischen dem drittem und dem vierten Text. Im dritten Text ist der erste Teil nicht vorhanden. Hier kann man sehen, dass der Teil welcher im vierten Text hinzugefügt wurde nur in der oberen Zeile steht.

Abbildung 4.3: Direkter Textvergleich zwischen jeweils zwei Texten auf dem Beispieldatensatz.

Abbildung 4.4: Corpora Comparison zwischen den ersten beiden Texten zum Rest des Beispieldatensatzes. Wörter in Gold repräsentieren die ersten beiden Texte.

der unteren Zeile in Cyan markiert. Das würde sich lediglich darin unterscheiden, welche der beiden Kanten markiert wurde. In diesem Beispiel tritt jedes Szenario genau einmal auf, in der Realität können bei jedem Textvergleich alle Szenarien mehrfach vorkommen.

4.3 Textgruppen

In der Adjazenzmatrix kann es vorkommen, dass einige Zeitungsartikel hervorstechen. So können diese Artikel untereinander stärker verwandt sein. Einige Texte können so Wörter enthalten, welche sie von anderen Texten unterscheidet. Dazu kann man einzelne Texte zu einer zu untersuchenden Gruppe hinzufügen. Diese Gruppe wird dann mit den übrigen Texten verglichen. Unser Prototyp sucht dann Wörter, welche die markierte Gruppe widerspiegeln, sowie Wörter, welche den Rest der nicht markierten Wörter repräsentieren. Wird das Datum eines Textes markiert, so ist auf der rechten Seite zu sehen, durch welche Wörter sich diese beiden Gruppen unterscheiden. Je wichtiger dieses Wort ist, das die beiden Gruppen unterscheidet, desto weiter vorne steht es in der Liste der Wörter. Zusätzlich ist abzulesen, zu welchem der beiden Gruppen dieses Wort gehört. Ist das Wort in Gold geschrieben, so gehört es zu der Gruppe, welche markiert ist. Hat das Wort eine blaue Farbe so gehört dieses Wort eher zum Rest, welcher nicht markiert ist.

In Abbildung 4.4 ist eine Wortliste einer Textgruppe zu sehen. Hier sind die ersten beiden Texte markiert, demnach gehören die zwei identischen Texte zu der Gruppe, die untersucht wird. Den Rest bildet ein Text, welcher sich nicht mit diesen überschneidet sowie die Konkatenation dieser beiden Texte. Wörter, die zu Unterscheidung der beiden Gruppen relevant sind, stehen in der Liste weiter vorne.

Auffällig sind hier die Wörter „something“, „completly“ und „different“. Diese Wörter kommen in den markierten Texten nicht vor und repräsentieren so die Artikel, welche nicht markiert sind. Die anderen Wörter sind zusätzlich in dem vierten Text vorhanden. Demnach grenzen diese Wörter weniger diese Gruppe von der anderen ab. Erkennbar ist dennoch, dass sie in den markierten Texten häufiger vorkommen als im Rest des Datensatzes.

5 Implementierung

In diesem Kapitel beschreiben wir, wie wir die Konzepte aus Abschnitt 4 ausgeführt haben. Damit wir für die Visualisierung eine einheitliche Datenstruktur verwenden können, ohne die zu analysierenden Daten stark zu limitieren, haben wir uns bei der Visualisierung dafür entschieden, dass, bevor die Daten angezeigt werden, eine Vorberechnung stattfinden muss. Wir gehen davon aus, dass jeder Datensatz D aus n Tupeln besteht.

$$D = \{(t_i, d_i)\}_{i=1}^{n}$$

Dabei ist t_i das Veröffentlichungsdatum des Textes und d_i der Text mit diesem Veröffentlichungsdatum. Das Aufspalten in eine Vorberechnung und eine Visualisierung bringt uns eine interaktive Visualisierung mit geringeren Wartezeiten.

5.1 Vorberechnungen

Die Datensätze erhalten wir im csv-Format und exportieren diese als json-Datei. Die Datensätze kommen als Tabelle, bei welchem jede Zeile in der Tabelle für einen Artikel steht. In der ersten Spalte steht jeweils der Inhalt des Artikels und in der zweiten Spalte steht das Datum der Veröffentlichung des Artikels.

Diese Daten benötigen wir als Graph, in welchem die Artikel die Knoten des Graphen darstellen und die paarweisen Textvergleiche die Kanten des Graphen. Dazu vergeben wir dem Knoten zunächst eine eindeutige ID und speichern diese Informationen in einer Knotenliste. Die Zeit, die wir dafür benötigen, können wir für n Artikel mit $\mathcal{O}(n)$ abschätzen. Für die Kanten benötigen wir den paarweisen Vergleich aller Knoten. Daraus folgt, dass wir für n Artikel die folgenden Berechnungen n^2 mal ausführen. Für jede Kante berechnen wir einmal den Jaccard-Koeffizienten für alle Zeichen-N-Gramme bis zu einer Länge von 30 Zeichen. Für die Berechnung dieser Koeffizienten benötigen wir – bei einer Textlänge von m – eine Laufzeit, welche in $\mathcal{O}(m)$ liegt. Außerdem berechnen wir die LCS auf jeder Kante, was zu einer Laufzeit in $\mathcal{O}(m^2)$ führt. In diesem Schritt speichern wir noch einen Hilfstext, mit welchem wir die Visualisierung des direkten Textvergleiches beschleunigen können. Zusätzlich benötigen wir eine Datenstruktur, um die *Corpora Comparison*, wie in Abschnitt 4.3 beschrieben, schnell zu berechnen. Dazu erstellen wir für den gesamten Datensatz ein Wortlexikon für jedes Dokument. Um dieses Lexikon zu erstellen, benötigt man für jedes Wort und jedes Dokument eine Operation. Allgemein lässt sich damit die Laufzeit dieser Vorberechnungen mit $\mathcal{O}(n^2 \cdot m^2)$ abschätzen.

5.2 Adjazenzmatrix

Einen Überblick über den gesamten Datensatz bildet die Adjazenzmatrix. Diese präsentiert den Graphen, welcher in der Vorberechnung erstellt wurde. Zwischen zwei gleichen Knoten bietet ein Textvergleich keinen Vorteil, aus diesem Grund belassen wir die Diagonale der Adjazenzmatrix leer. Da es sich bei dem Graphen um einen ungerichteten Graphen handelt, sind die Kanten zwischen zwei gleichen Knoten in beide Richtungen identisch. Aus diesem Grund können beide Textvergleiche, der Jaccard-Koeffizient und die Überdeckung der LCS zwischen zwei Texten in einer gemeinsamen Ansicht angezeigt werden. Wir trennen die Kanten mit der Diagonalen der Matrix, welche keine Einträge besitzt.

Für jede Kante bestimmen wir eine Farbe, welche die Kante und den entsprechenden Textvergleich repräsentiert. Unterhalb der Diagonalen zeigen wir den Wert der Überdeckung der LCS an. Dafür nehmen wir die Ordnung der Überdeckung der Teilsequenz mit dem kürzeren Text. Diese Ordnung skalieren wir linear auf einen Wert zwischen 0 und 255, dies bildet den Blauwert im RGB-Farbraum. Die anderen beiden Einträge sind 0, so bekommen wir eine Abstufung der Einträge von blau zu schwarz. Diese Kante wird als quadratisches svg-Rechteck in dieser Farbe an der entsprechenden Stelle hinzugefügt. In der anderen Hälfte der Kanten skalieren wir den Jaccard-Koeffizient, welcher einen Wert zwischen 0 und 1 annimmt, wieder linear von 0 bis 255. Dieser ist der Rot-Wert, so ist oberhalb der Diagonalen der Jaccard-Koeffizient in roter Farbe kodiert.

Um diese Kanten auf der Matrix zu verteilen, benötigen wir bei n Knoten im Datensatz n^2 Kanten. Diese werden auf dem Bildschirm beim Starten der Visualisierung angezeigt. Der Aufwand dafür liegt in $\mathcal{O}(n^2)$. Der Nutzer kann auf jede Kante klicken, welche nicht auf der Diagonalen liegt. Dann wird diese Kante sowie dieselbe Kante in die entgegengesetzte Richtung mit einer Umrandung markiert. Außerdem öffnet sich der Textvergleich, wie in Abschnitt 4.2 beschrieben.

5.3 Knoten

Die Knoten des Graphen zeigen wir oberhalb sowie links von der Adjazenzmatrix an. Diese Knoten sind nach ihrem Datum sortiert. Dabei ist der Knoten mit dem frühesten Datum sowohl einmal am linken Ende, als auch einmal am oberen Ende der Knotenliste zu finden. Der Hintergrund des Datums kann vier verschiedene Farben annehmen. Zu Beginn hat jeder Knoten einen transparenten, beziehungsweise weißen Hintergrund. Wir nehmen an, dass jede Kante einen Start und einen Zielknoten hat, wir zeigen dann auf der linken Seite die Startknoten und oben an der Matrix die Zielknoten an. Ein Knoten, der in der oberen Zeile angezeigt wird, kann zusätzlich noch einen Hintergrund in Cyan haben. Wird eine Kante markiert, so ist dieser Knoten oberhalb der Matrix der Zielknoten der Kante. Zur selben Zeit wird auf der linken Seite der Startknoten mit einem violetten Hintergrund eingefärbt.

Außerdem ist es möglich, auf einen Knoten zu klicken. Dann wird dieser Text zu einer Gruppe von Texten hinzugefügt. Diese Gruppe wird dann – wie in Abschnitt 4.3 beschrieben

– analysiert. Ist ein Knoten dieser Gruppe hinzugefügt worden, so bekommt dieser einen goldenen Hintergrund.

Beim Starten der Visualisierung werden diese Knoten horizontal sowie vertikal an die Matrix angrenzend dargestellt. Die Zeit, die wir benötigen, um diese Knoten auf dem Bildschirm anzuzeigen, ist abhängig von den Knoten n aus dem Datensatz. Das Anzeigen der Knoten beim Starten der Visualisierung liegt demnach in $\mathcal{O}(n)$.

5.4 Darstellung von Texten mit gleicher Veröffentlichung

Zwischen den einzelnen Knoten und der Adjazenzmatrix ist ein kleiner Zwischenraum. Dieser ist bewusst gewählt, um Texte, welche am selben Tag veröffentlicht wurden, besonders hervorzuheben. So kann der Nutzer auf einen Blick sehen, welche Texte am selben Tag veröffentlicht wurden. Diese Hervorhebungen sind hellgraue Rechtecke, welche am selben Tag veröffentlichte Texte verbindet.

5.5 Textgruppen

Die Texte können – wie oben beschrieben – zu einer Gruppe hinzugefügt werden. Sobald ein Text zu dieser Gruppe hinzugefügt oder aus der Gruppe entfernt wird, so wird die Ansicht auf der rechten Seite aktualisiert. Wie in Abschnitt 2.4 berechnen wir mithilfe des Ansatzes von Rayson und Garside [RG00] die relevantesten Wörter, um die beiden Textgruppen zu unterscheiden. Dies führt zu einer Liste an Wörtern, welche wir den beiden Gruppen zuordnen können. Da in Texten sehr viele Wörter vorkommen, die die Themen der Texte nicht widerspiegeln, können wir diese wie folgt herausfiltern. Bevor wir die Wörter anzeigen, entscheiden wir für jedes Wort, ob es zu einem Wort auf einer schwarzen Liste steht. Dabei nutzen wir nicht nur eine Liste der gängigsten englischen *stopwords* sondern ebenso auch eine Liste von deren häufig auftretenden OCR-Fehlern. Kommt ein Wort durch diesen Filter, so wird es zu den Wörtern in der entsprechenden Farbe hinzugefügt. Da die Wörter nach ihrer Relevanz sortiert sind, stehen relevantere Wörter am Anfang. Beim Klicken auf einen Knoten wird diesem eine goldene Farbe zugewiesen. Ebenso bekommt derselbe Knoten auf der entsprechend anderen Seite der Matrix auch eine goldene Färbung.

Da hier maximal alle Knoten markiert sein können, liegt die Komplexität zum Markieren dieser n Knoten in $\mathcal{O}(n)$. Um die relevanten Wörter aus den Textgruppen zu finden, müssen wir jedes Wort eines jeden Dokumentes betrachten. Das führt bei n Knoten mit jeweils m Wörter zu einer Komplexität, die in $\mathcal{O}(n \cdot m)$ liegt.

5.6 Textdifferenz

Klickt der Nutzer auf eine Zelle in der Matrix, so ist diese Kante mit zwei Knoten verbunden. Der Textinhalt dieser Knoten wird dann auf der rechten Seite angezeigt. Abschnitte, welche in beiden Texten also auch in der berechneten Teilsequenz vorkommen, werden in Hypertext Markup Language (HTML) als Text dargestellt. Ein Text, welcher in beiden

Texten unterschiedlich ist, wird in zwei Container aufgeteilt, wovon der obere der Container den Text in Cyan und der untere den Abschnitt des anderen Textes entsprechend in Violett darstellt.

Um diese Texte zu erstellen, muss jeder Text einmal vollständig abgebildet werden. Dies hat bei einer Textlänge von m eine Komplexität in $\mathcal{O}(m)$.

Dieser Textvergleich wird ausgeführt, sobald eine Kante in der Matrix ausgewählt wurde. Dabei werden zusätzlich noch die damit verbundenen Knoten in den entsprechenden Farben markiert. Dazu suchen wir in der Liste der Knoten die verbundenen Knoten heraus. Deshalb steigt in unserer Implementierung für n Knoten linear zu dieser Zahl auch die Zeit, um diese Knoten hervorzuheben. Außerdem markieren wir die Kante, welche – an der Diagonalen gespiegelt – zu den gleichen Knoten gehört. Mit dieser Visualisierung kann der Nutzer auf einen Blick die beiden Metriken ablesen. In unserer Implementierung bedeutet dies, dass wir jede Kante einmal durchsuchen, um zu sehen, ob diese markiert werden muss. Dies liegt bei n Knoten und entsprechend n^2 Kanten in $\mathcal{O}(n^2)$.

5.7 Zurücksetzen

Die Visualisierung kann wieder auf den Ausgangszustand zurückgesetzt werden. Dazu kann der Nutzer auf den „Reset“ Knopf klicken, welcher links oben an die Adjazenzmatrix angrenzt.

Bei jedem Hinzufügen zur initialen Visualisierung werden die Attribute, die in HTML hinzugefügt werden, zu einer entsprechenden Klasse zugeordnet. Über diese Klassen können diese Objekte einfach wieder vom Bildschirm entfernt werden. Beim Anklicken dieser Schaltfläche werden alle Objekte, die diesen Klassen zugeordnet sind und somit nicht zur initialen Visualisierung gehören, entfernt. Hierbei kann die Anzahl dieser Objekte abgeschätzt werden. Es kann jeder Knoten markiert sein, falls alle Knoten zur gewählten Gruppe hinzugefügt worden sind. Es können maximal zwei Kanten markiert sein. Außerdem können auf der rechten Seite der Visualisierung alle Wörter der Texte aufgelistet sein. Daraus folgt für das Zurücksetzen eine Komplexität, die in $\mathcal{O}(n \cdot m)$ liegt.

5.8 Anpassungen für reale Datensätze

Bei realen Datensätzen treten OCR-Fehler auf, die wir besonders in den detaillierteren Ansichten vermeiden möchten. So kommt es vor, dass ein Zeichen falsch erkannt wurde und darum in beiden Texten als identisch angesehen wird. Um dies zu vermeiden, teilen wir kurze Abschnitte, welche maximal zwei Zeichen lang sind, auf die beiden Zwischenzeilen auf. So fallen lange Abschnitte stärker ins Gewicht und werden im direkten Textvergleich besser wahrgenommen.

Wir modellieren einen Text als Aneinanderreihung seiner Wörter. Das bedeutet, wir verwerfen alle Zeichen, welche nicht zu Wörtern gehören. Damit entfernen wir auch Kommata und Punkte im Text. Daraus folgt, dass wir Informationen verwerfen, die auf diesen Zeichen beruhen. Dies führt dazu, dass wir längere zusammenhängende Sequenzen von Wörtern

erkennen, sollte der Autor lediglich Kommata entfernt oder hinzugefügt haben. Im Normalfall ändert sich dadurch der Inhalt der Texte nicht erheblich, bringt uns jedoch deutlich weniger OCR-Fehler im Text. Aus diesem Grund haben wir uns für diese Vereinfachung entschieden.

Außerdem sind die Texte, auf welchen die Visualisierung angewandt werden soll, im Normalfall in englischer Sprache verfasst. Da wir die Setzung der Punkte im Text wie beschrieben entfernen, können wir demnach auch die Großschreibung für die Visualisierung ignorieren. Damit haben wir den Vorteil, dass wir besonders bei dem Vergleich der beiden Textkörper bei gleichen Wörtern auch eine Übereinstimmung haben. Diese ist unabhängig von der Großschreibung, da diese die Bedeutung des Wortes, vor allem in englischer Sprache, nicht ändert.

6 Evaluation

In diesem Kapitel zeigen wir auf, welche Einschränkungen unser Prototyp hat und welche Analysen mit unserem Visualisierungstool möglich sind. Dies stellen wir zum einen an einem Datensatz vor, den wir analysieren. Zum anderen haben wir noch einen Datensatz von einem Experten analysieren lassen, um damit unsere Visualisierung bewerten zu können.

6.1 Skalierbarkeit

Zur Bewertung des Prototyps diskutieren wir, wie sich dieses im Hinblick auf die Größe des Datensatzes verhällt.

6.1.1 Berechnungskomplexität

Die Komplexität der Vorberechnung liegt – wie bereits diskutiert – für n Artikel der Länge m in $\mathcal{O}(n^2 \cdot m^2)$. In der Tabelle 6.1 ist aufgelistet, wie lange diese Vorberechnungen für die von uns getesteten Datensätze dauern. Die Datensätze wurden auf einem Intel i7-3770 Prozessor mit 3,9 GHz auf jeweils 8 Threads und mit 16 GB Arbeitsspeicher berechnet. Beim Starten der Visualisierung müssen alle Kanten und Knoten aus der Datei eingelesen und angezeigt werden. Dies führt – wie in Abschnitt 5.2 beschrieben – zu einer Komplexität in $\mathcal{O}(n^2)$. Mit der Anzahl der Artikel steigt demnach auch der Aufwand, die Visualisierung anzuzeigen, deutlich.

Klickt der Nutzer auf einen Knoten, so startet der Vergleich der beiden Textkörper. Wir haben in Abschnitt 5.5 bereits beschrieben, wie sich der Rechenaufwand dafür abschätzen lässt. Dieser liegt bei n Artikeln mit der Länge m in $\mathcal{O}(n \cdot m)$.

Tabelle 6.1: Die folgende Tabelle zeigt die Vorberechnungszeit unserer Datensätze. Hierfür ist die Länge der einzelnen Artikel sowie die Anzahl der Artikel ausschlaggebend.

Datensatz	Anzahl der Artikel	Maximale Artikellänge	Zeit der Vorberechnung
Civil war	47 Artikel	1200 Zeichen	56 s
Krakatoa	51 Artikel	2600 Zeichen	11 min 10 s
Slavery	68 Artikel	4300 Zeichen	9 min 10 s
Maine	69 Artikel	2000 Zeichen	4 min 49 s
Ship explosion	125 Artikel	4200 Zeichen	1 h 5 min
Krakatoa large	199 Artikel	3200 Zeichen	6 h 30 min

Tabelle 6.2: Die folgende Tabelle zeigt, wie lang unsere Datensätze in der Visualisierung benötigen. Die Werte sind extern gemessen und aus diesem Grund nur als Richtwert zu verstehen.

Datensatz	Größe	Adjazenzmatrix	Kante auswählen	Knoten auswählen
Civil war	47 Artikel	0.5 s	0.2 s	0.6 s
Krakatoa	51 Artikel	0.5 s	0.2 s	0.6 s
Slavery	68 Artikel	0.8 s	0.2 s	0.8 s
Maine	69 Artikel	0.9 s	0.3 s	0.7 s
Ship explosion	125 Artikel	2.5 s	0.6 s	1.1 s
Krakatoa large	199 Artikel	8 s	0.7 s	2.5 s

Außerdem kann der Nutzer eine Kante auswählen. Damit wird der paarweise Textvergleich angezeigt. Dieser Aufwand liegt theoretisch in $\mathcal{O}(m)$. In unserer Implementierung hat dieses Verhalten jedoch einen Aufwand von $\mathcal{O}(\max(m, n^2))$. Dies haben wir in Abschnitt 5.6 beschrieben. Das Markieren der Kante auf der entsprechend anderen Seite der Diagonalen sollte in $\mathcal{O}(1)$ liegen.

Wir haben auf einigen Datensätzen gemessen, wie sich dies in der Praxis auswirkt. Dabei ist jedoch zu beachten, dass diese Zeiten nicht innerhalb des Programmes berechnet wurden, sondern extern gemessen wurden. Aus diesem Grund sind die Messwerte als Richtwert zu sehen.

In der Tabelle 6.2 sind diese Werte abzulesen. Dabei lief der Server sowie der Client auf demselben Rechner, um die Netzwerk-Latenz aus der Messung zu eliminieren. Der Rechner, auf welchem die Visualisierung lief, hat einen Intel i5-7200U, welcher mit 3,1 GHz taktet und mit 8 GB Arbeitsspeicher ausgestattet ist.

6.1.2 Visualisierungskonzept

Unser Konzept, mit welchem wir die Daten anzeigen, gibt uns Einschränkungen, wie groß die Datensätze sein können.

Auf einem Bildschirm mit einer Auflösung von 1920 Pixeln in der Breite und 1080 Pixeln in der Höhe sind Datensätze mit bis zu 90 Artikeln noch gut sichtbar. Hier bleiben für jeden Knoten der Matrix noch 10 Pixel in der Höhe beziehungsweise 10 Pixel in der Breite. Demnach sind auch die Quadrate, auf welche der Nutzer klicken kann, mit einer Seitenlänge von 10 Pixeln noch gut nutzbar.

Bei einem Testdatensatz mit 200 Artikeln bleiben jedem Eintrag in der Matrix noch 3 Pixel. Damit bleiben einzelne Matrixeinträge noch unterscheidbar. Ein Text, welcher den Knoten mit dem Datum beschreibt, wird jedoch nicht mehr lesbar. Außerdem wird die Interaktion deutlich schwerer, da das Treffen eines Quadrates mit einer Seitenlänge von 3 Pixeln sehr mühsam ist.

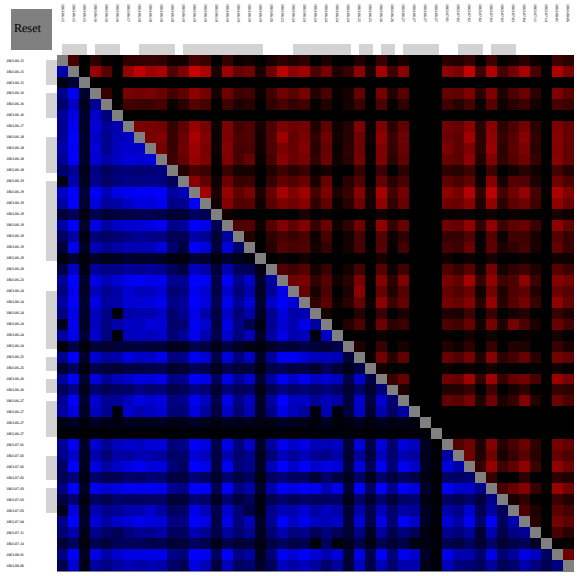


Abbildung 6.1: Diese Abbildung zeigt den Überblick über unseren Datensatz zum Bürgerkrieg. Auffällig sind zwei Texte, welche am selben Tag veröffentlicht wurden, wobei der Jaccard-Koeffizient klein ist und die LCS eine geringe Überdeckung hat.

6.2 Visualisierung eines realen Datensatzes

Zur Evaluierung des Prototyps werden wir einen historischen Datensatz aus einer Sammlung betrachten. Außerdem haben wir eine Literaturwissenschaftlerin auf kleineren Datensätzen von 10 Artikeln unseren Prototyp testen lassen. Damit bekommen wir eine weitere Bewertung unseres Prototyps.

6.2.1 Visualisierung eines historischen Datensatzes

Im Folgenden schauen wir uns einen Datensatz mit Artikeln zu einem Bürgerkrieg an. Dieser Datensatz stammt aus einer Sammlung von historischen Zeitungsartikeln, aus dem Projekt *Oceanic Exchanges: Tracing Global Information Networks in Historical Newspaper Repositories* [Oce17]. Dabei wurde in dieser Sammlung eine Reprinting-Analyse ausgeführt, sodass wir eine Menge von 50 bis 90 ähnlichen Artikeln erhalten.

In Abbildung 6.1 ist der Überblick über den gesamten Datensatz zu sehen. Auf den ersten Blick fallen im rechten, roten Bereich die Spalten und Reihen auf, welche im Vergleich einen sehr geringen Jaccard-Koeffizient haben. Einige dieser Spalten haben auch im blauen Bereich eine geringe Überdeckung der größten Teilsequenz.

Des weiteren kann man sehen, dass die ersten drei Texte am selben Tag veröffentlicht wurden, da sie mit einem grauen Rechteck verbunden sind. Zwischen diesen ersten drei Texten ist die Ähnlichkeit relativ gering, da in der gesamten Spalte und der gesamten Zeile an diesen Knoten die Kanten durchgehend schwarz sind. Außerdem sind von rechts aus im letzten Drittel am 27.06.1863 drei Texte verfasst worden, welche – verglichen mit den

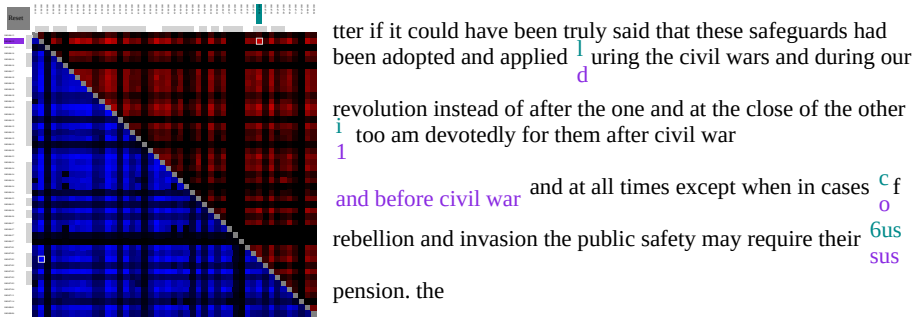


Abbildung 6.2: Im Datensatz zu einem Bürgerkrieg ist eine Kante ausgewählt. Zu sehen ist ein hinzugefügter Textabschnitt und vier OCR-Fehler.

anderen Texten – einen geringen Jaccard-Koeffizienten haben. Davon hat der erste Text im Vergleich zu den anderen Texten eine hohe Überdeckung der Teilsequenz.

Im Allgemeinen kann man folgenden Verlauf erkennen: Zu Beginn haben die Texte eine hohe Überdeckung, im Laufe der Zeit kommen jedoch mehr Texte hinzu, die sich wahrscheinlich mit anderen Aspekten beschäftigen. Dies könnte zur Folge haben, dass mehr Texte mit deutlich geringerer Überdeckung hinzukommen.

Außerdem ist auffällig, dass zu Beginn erst wenige Artikel am selben Tag erschienen sind. Darauf folgen zwei Tage, an welchen mehr als fünf Artikeln verfasst wurden. Danach werden Artikel am selben Tag jedoch wieder seltener.

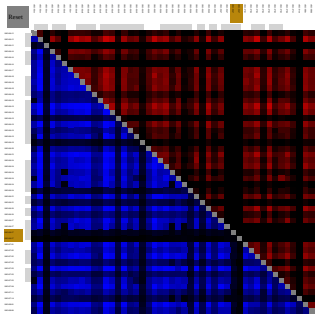
Im direkten Vergleich von zwei Texten kann man in Abbildung 6.2 erkennen, dass im zweiten Text, welcher violett markiert ist, der Teil „and before civil war“ hinzugekommen ist. Zu sehen sind auch zwei OCR-Fehler. Zum einen wurde das “d“ in „during“ im ersten, cyanfarbenen Text als „l“ erkannt zum anderen wurde das „i“ im zweiten Text fälschlicherweise als „1“ erkannt.

In Abbildung 6.3 sind die zwei Texte markiert, bei welchen der Jaccard-Koeffizient sowie die längste Teilsequenz bei allen anderen Texten sehr niedrig sind. Dabei kann man an den Wörtern, welche auf der rechen Seite stehen, erkennen, welche Themen bei diesen Texten thematisiert werden. In Gold stehen die Wörter „devoted“, „power“, „protection“ und „demonstrate“, dies könnte bedeuten, dass in diesen Texten besonders politische Aspekte angesprochen werden.

6.2.2 Visualisierung kleinerer Datensätze aus der Literaturwissenschaft

In der Literaturwissenschaft läuft aktuell ein Projekt, in welchem Zeitungsartikel aus dem 19. Jahrhundert aus den USA – welche in deutscher Sprache verfasst wurden – betrachtet werden sollen. Aus einem großen Datensatz werden mittels Reprint-Analysen kleine Cluster von fünf bis zehn Texten gebildet. Diese Texte werden dann weiter auf ihre Änderungen untereinander verglichen, dabei soll unser Prototyp diese Aufgabe erleichtern.

Wir haben unseren Prototyp einer Literaturwissenschaftlerin, die an diesem Projekt arbeitet, zur Verfügung gestellt. Dabei haben wir beobacht, wie sie diesen Prototyp annimmt, indem wir ihre Interaktion mit dem Prototyp aufgezeichnet haben. Damit versuchten wir zu



his, aod, tbe, proceed, devoted, utter, ill, jury, arbitrary, power, apparently, trial, resolve, rights, intended, against, especially, commotion, rebellion, protection, demonstrate, proposition, instead, require, invasion, may, all, their, devotedly, demonstration, secured, suspension, safety, cases, tho, not, for, aud, case, under, bad, safe, tion, stood, resolutions, tbo, limes, clone, luring, cured, afltr, dote, yeara, sion, hav, more, tell, test, six, which, while, government, ter, revolutionwould, tbem, plied,

Abbildung 6.3: Im Datensatz zu einem Bürgerkrieg sind zwei auffällige Knoten markiert, welche mit dem Rest verglichen werden. In Gold stehen unter anderem die Wörter: „devoted“, „protection“ und „demonstrate“. Diese Wörter reräsnetieren den ausgewählten Textkörper, der aus zwei Knoten besteht.

erkennen, wie unser Prototyp für diese Anwendung geeignet ist. Sie hatte dabei drei Datensätze mit fünf bis zehn Artikeln aus ihrer Arbeit verwendet.

Als erste Interaktion klickte sie auf die Zelle in der Matrix, diese zeigte den höchsten Jaccard-Koeffizienten von allen Zellen an. Anschließend betrachtete sie Texte, welche kleine Jaccard-Koeffizienten besitzen. Dieses Verhalten änderte sich auch bei neuen Datensätzen nicht. Dies könnte bedeuten, dass diese Kanten auch interessant sind und diese die erwarteten Informationen beinhalten.

Nachdem sie den Prototyp für eine Stunde benutzt hatte, stellte sich heraus, dass überwiegend Zellen in der Adjazenzmatrix angeklickt wurden. Diese Kanten wurden im Durchschnitt zehn Sekunden betrachtet, bevor sie die nächste Kante angeklickte. Dies könnte daran liegen, dass die Kante nicht den gewünschten Inhalt bot, oder nur überprüft wurde, ob die Kante den erwarteten Inhalt aufweist. Während diesem Test wurden Knoten nur selten oder nach Aufforderung angeklickt. Diese bieten womöglich zu wenig relevante Informationen oder die Informationen werden zu abstrakt als Textliste angezeigt. Jedoch sind diese Cluster aus dem kleinen Datensatz sehr eng verwandt, so könnte diese Information auf diesen Clustern bereits bekannt sein.

Aus der Matrix sowie der paarweisen Textdifferenz konnte die Literaturwissenschaftlerin Hypothesen aufstellen, welche vorher nicht aufgefallen waren. So können mit diesem Prototyp Verlage erkannt werden, von welchen die Artikel eine deutlich höhere Quote an OCR-Fehlern aufweist. Zudem erkannte sie in der Textdifferenz schnell, wenn Wörter amerikanisiert wurden. Das bedeutet, dass in einem Wort Konsonanten so abgeändert werden, wie sie im englischen ausgesprochen werden.

Für ihre Arbeit wären noch zusätzliche Funktionen hilfreich. Es wäre sinnvoll, wenn zu den einzelnen Artikeln weitere Informationen verfügbar wären. Weitere Angaben zum Ort der Veröffentlichung sowie zum Verlag der Artikel wären für sie ebenfalls nützlich.

6.3 Auswertung

Mit unserem Prototyp kann der Nutzer einen Überblick über einen Datensatz erhalten und Hypothesen aufstellen, welche auch durch Lesen aller Texte nicht auffallen würden. Unser Prototyp hat jedoch auch Einschränkungen, so sind Datensätze mit mehr als 100 Artikeln für unsere Visualisierung nicht geeignet. Für den weiteren Einsatz dieses Prototyps ist es denkbar, zusätzliche Funktionen hinzuzufügen.

7 Zusammenfassung

In diesem Kapitel fassen wir die Vorgehensweise der Arbeit zusammen und beschreiben Ideen, mit welchen der Prototyp weiterentwickelt werden kann.

7.1 Fazit

In dieser Arbeit haben wir einen Prototyp zur Analyse von Textsequenzen erstellt. Mit diesem soll es unter anderem Forschern der Literaturwissenschaft erleichtert werden, eine Menge an Artikeln zu untersuchen. Damit ist es möglich, Erkenntnisse zu erlangen, welche sonst mühsam erarbeitet werden müssten. Wir können mit diesem Prototyp dem Nutzer das Lesen jedes einzelnen Textes ersparen, falls die Texte im Datensatz auf ihre Ähnlichkeit untersucht werden sollen.

Da wir alle Texte zur selben Zeit anzeigen möchten, benötigen wir Metriken mit welchen wir die Texte jeweils paarweise vergleichen können. Diese Metriken haben wir im Kapitel 2 in den Grundlagen beschrieben. Wir sammeln alle paarweisen Vergleiche und stellen sie in der Adjazenzmatrix dar. Mit dieser Matrix kann der Nutzer interagieren. Der Nutzer kann sich die Textdifferenz zwischen zwei Texten anzeigen lassen oder einige Texte mit den übrigen Texten vergleichen. Die Darstellung ist im Kapitel 4 näher aufgeführt. Während wir den Prototyp entwickelt haben, sind wir besonders bei Wartezeiten in der Interaktion auf Herausforderungen gestoßen. Hier ist besonders die Laufzeit der Interaktion, sowie der Überblick über markierte Teile der Visualisierung zu nennen. Wie wir diese gelöst haben, haben wir im Kapitel 5 geschildert. Im Kapitel 6 haben wir erläutert, wie sich der Prototyp auf einigen Datensätzen verhält. Außerdem haben wir dort diskutiert, wie sich der Prototyp für größere Datensätze verhalten wird.

Wir wollten einen Prototyp entwickeln, welcher für viele verschiedene Datensätze anwendbar ist. Darum haben wir uns beim Erstellen des Prototyps bewusst dafür entschieden, dass unser Prototyp nicht auf eine Sprache festgelegt ist. Da wir keine Sprachverarbeitung für bestimmte Sprachen nutzen, kann unsere Visualisierung auch für andere Sprachen verwendet werden. Ebenso ist es zum Beispiel möglich, den Verlauf einer Datei innerhalb eines Git Repositories zu analysieren.

Dennoch bietet der Prototyp Einschränkungen in der Größe der zu visualisierenden Daten. Wie bereits im Kapitel 6 beschrieben, bietet dieser bei gängigen Bildschirmen keine guten Ergebnisse für mehr als 90 Knoten im Datensatz.

7.2 Ausblick

Der Prototyp ist noch nicht vollständig ausgereift. Auf großen Datensätzen mit 200 Knoten dauert das Updaten der Ansicht ca. zwei Sekunden. Diese Zeit könnte noch minimiert werden, da das Markieren von Knoten und Kanten bei uns deutlich länger dauert als es nötig ist. Für kleinere Datensätze ist diese Zeit jedoch deutlich kleiner und schränkt die Interaktion nicht weiter ein.

Unser Prototyp wurde mit englisch sprachlichen Datensätzen entwickelt. Aus diesem Grund hat der Prototyp bisher noch keine Unterstützung für deutsche Umlaute. Diese werden in der Visualisierung übersprungen und müssten bei deutschen Datensätzen in Betracht gezogen werden. Um den Prototyp darüberhinaus weiterzuentwickeln, stehen folgende Möglichkeiten an: Bisher musste der Wert k, welcher für die Zeichen-N-Gramme verwendet wird, gewählt werden, bevor der Datensatz eingesehen werden kann. Es wäre möglich, diesen Wert ebenfalls interaktiv zur Laufzeit zu ändern, so kann der Nutzer diesen auf den Datensatz anpassen. Die Überdeckung der LCS wird bisher strikt so geordnet, dass keine zwei Kanten die identische Ordnung haben können. Dies kann zu einer Fehlinterpretation führen, falls ein Großteil der Texte identisch ist. Diese haben alle eine vollständige Überdeckung, werden jedoch mit einem unterschiedlichen Wert angezeigt. Außerdem ist das Wechseln des Datensatzes umständlich. Falls der Nutzer diesen häufig ändert, sollte dies in den Prototyp integriert werden. Aktuell muss der Nutzer den Dateipfad des Datensatzes im Programmcode ändern.

Literaturverzeichnis

[GM70] A. J. Gibbs und G. A. Mcintyre. „The Diagram, a Method for Comparing Sequences“. In: *European Journal of Biochemistry* 16.1 (1970), S. 1–11. ISSN: 1432-1033. DOI: 10.1111/j.1432-1033.1970.tb01046.x (zitiert auf S. 11, 12).

[HFM07] N. Henry, J.-D. Fekete und M. J. McGuffin. „NodeTrix: a Hybrid Visualization of Social Networks“. In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (Nov. 2007), S. 1302–1309. ISSN: 2160-9306. DOI: 10.1109/TVCG.2007.70582 (zitiert auf S. 3, 13).

[Hir77] D. S. Hirschberg. „Algorithms for the Longest Common Subsequence Problem“. In: *Journal of the ACM (JACM)* 24.4 (Okt. 1977), S. 664–675. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/322033.322044 (zitiert auf S. 6).

[HLLE14] F. Heimerl, S. Lohmann, S. Lange und T. Ertl. „Word Cloud Explorer: Text Analytics Based on Word Clouds“. In: *2014 47th Hawaii International Conference on System Sciences*. 2014 47th Hawaii International Conference on System Sciences. ISSN: 1530-1605. Jan. 2014, S. 1833–1842. DOI: 10.1109/HICSS.2014.231 (zitiert auf S. 14).

[Jac12] P. Jaccard. „The Distribution of the Flora in the Alpine Zone.1“. In: *New Phytologist* 11.2 (1912), S. 37–50. ISSN: 1469-8137. DOI: 10.1111/j.1469-8137.1912.tb05611.x (zitiert auf S. 8).

[JGBS15] S. Janicke, A. Geßner, M. Buchler und G. Scheuermann. „Visualizations for Text Re-use“. In: *2014 International Conference on Information Visualization Theory and Applications (IVAPP)* (Okt. 2015), S. 12 (zitiert auf S. 3, 11, 12).

[MOB+12] J. P. McIntire, O. I. Osesina, C. Bartley, M. E. Tudoreanu, P. R. Havig und E. E. Geiselman. „Visualizing weighted networks: a performance comparison of adjacency matrices versus node-link diagrams“. en. In: Baltimore, Maryland, Mai 2012, 83891G–83891G–7. DOI: 10.1117/12.920012 (zitiert auf S. 6).

[Oce17] Oceanic Exchanges Project Team. *Oceanic Exchanges: Tracing Global Information Networks In Historical Newspaper Repositories*. 2017. URL: http://oceanicexchanges.org/ (besucht am 10. 05. 2020) (zitiert auf S. 29).

[Phi71] J. B. Phipps. „Dendrogram Topology“. In: *Systematic Zoology* 20.3 (Sep. 1971), S. 306. ISSN: 00397989. DOI: 10.2307/2412343. (Besucht am 12. 06. 2020) (zitiert auf S. 13).

[RG00] P. Rayson und R. Garside. „Comparing corpora using frequency profiling“. In: *Proceedings of the workshop on Comparing corpora - Volume 9*. WCC '00. Association for Computational Linguistics, Okt. 2000, S. 1–6. DOI: 10.3115/1117729.1117730 (zitiert auf S. 4, 9, 10, 23).

[TT13] G. Teschl und S. Teschl. *Mathematik für Informatiker*. eXamen.press. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN: 978-3-642-37971-0 978-3-642-37972-7. DOI: 10.1007/978-3-642-37972-7 (zitiert auf S. 5).

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift