Digital Journalism
Automatisierte Dossier-Erstellung mittels Text-Mining

Jan Paul Assendorp

Hamburg University of Applied Sciences, Dept. Computer Science,
Berliner Tor 7
20099 Hamburg, Germany
janpaul.assendorp@haw-hamburg.de

I. Einleitung

Im Journalismus werden Dossiers als Zusammenstellungen von Dokumenten verstanden, die sich mit dem gleichen thematischen Hintergrund beschäftigen. Dabei kann es sich um Dokumente jeglicher Form handeln, die sich thematisch in den verschiedensten Bereichen bewegen können. Geläufig sind Dossiers z. B. als Zusammenstellung aller verfügbarer journalistischer Artikel zu einem politischen oder historischen Ereignis, das für den Leser von Interesse ist.

Der Journalismus steht im digitalen Zeitalter vor der Herausforderung, die aufkommenden Trends aufzugreifen und zu verwerten, um auf den neuen Märkten bestehen zu können. Der Bereich *Digital Journalism* versucht dabei den Journalismus mit den Entwicklungen der Informatik-Domäne zu vereinigen. Dabei steht besonders die Verwertung von Text durch Mining-Verfahren im Fokus. Das Ziel ist hier, Prozesse des Journalismus zu automatisieren, oder den bearbeitenden Journalisten in seiner Arbeit zu unterstützen.

Text-Mining als Ausprägung des Data-Mining wird bereits seit einigen Jahren erfolgreich angewendet, um große Datenmengen in Textform verwerten zu können. Hinsichtlich der Algorithmen zur Clusteranalyse oder zur Klassifizierung von Dokumenten sind zahlreiche Verfahren bekannt und wurden über die Zeit optimiert. Die Herausforderung im Text-Mining liegt derzeit in der Datenrepräsentation von Texten, welche über domänenspezifische Informationen gewichtet und danach untereinander verglichen werden können.

In dieser Ausarbeitung sollen die thematischen Grundlagen des Text-Mining zusammengefasst und der aktuelle Stand der Forschung in Bezug auf das Verarbeiten von Textdokumenten aufgezeigt werden. Damit soll die thematische Grundlage für die zukünftige Projektarbeit erarbeitet werden. Es wird also besonders untersucht, wie der Prozess des Text-Mining hinsichtlich der automatisierten Erstellung von Dossiers anhand eines Dokumentenbestandes angewendet werden kann und an welchen Stellen eigene Entwicklungen gefordert sind.

Abschnitt II beschreibt zunächst den Aufbau eines Text-Mining-Prozess und zeigt zwei Algorithmen zur Clusteranalyse. Der darauf folgende Abschnitt III vergleicht zwei Tools zur Realisierung eines Text-Mining-Prozesses. Schließlich beschreibt Abschnitt IV den aktuellen Stand der Forschung anhand von vergleichbaren Arbeiten und gibt einen Ausblick auf die Problemstellung für die Projekt-Veranstaltung. Im Zuge des Ausblicks werden zudem Tools hinsichtlich einer für das Projekt zu verwendenden Tool-Chain betrachtet.

II. Text-Mining

Das *Text-Mining* ist die Anwendung von Data-Mining unter Verwendung von Daten in Textform. Das Ziel beim Text-Mining besteht darin, aus der Analyse von Textdokumenten einen Wissensgewinn zu erzielen. Dabei unterscheidet sich das Text-Mining vor allem in der Art der Strukturierung der zu verarbeitenden bzw. zu analysierenden Daten. Daten, die in Textform vorliegen gelten dabei im Sinne des Data-Mining als unstrukturierte Daten. Zwar folgt Text in der Regel einer gültigen Grammatik und damit auch einer allgemeinen Strukturvorgabe, allerdings lassen sich darauf ohne vorangehende Vorverarbeitung keine Algorithmen des Data-Mining zur Datenanalyse anwenden. In dieser Vorverarbeitung müssen zunächst relevante Inhalte erkannt und extrahiert werden. Dazu ist ggf. eine ausgeprägte Kenntnis der Fachdomäne oder der zugrundeliegenden natürlichen Sprache erforderlich. Durch die Vorverarbeitung werden Texte in eine geeignete Datenrepräsentation umgewandelt, welche im Folgenden zum Data-Mining verwendet werden kann. Der Prozess des Text-Mining kann in einzelne Prozessschritte unterteilt werden, die in Abbildung 1 gezeigt sind.

Im Abschnitt II-A werden zunächst die einzelnen Prozessschritte des Text-Mining beschrieben. Besonders relevant ist für diese Betrachtung das *Preprocessing* und das eigentliche Data-Mining. Aus diesem Grund wird das Preprocessing am Abschnitt II-B gesondert betrachtet. Das Data-Mining beschränkt sich durch den Anwendungsfall dieser Arbeit auf die Clusteranalyse. Dazu werden im Abschnitt II-C und im Abschnitt II-D zwei Algorithmen zur Clusteranalyse vorgestellt.

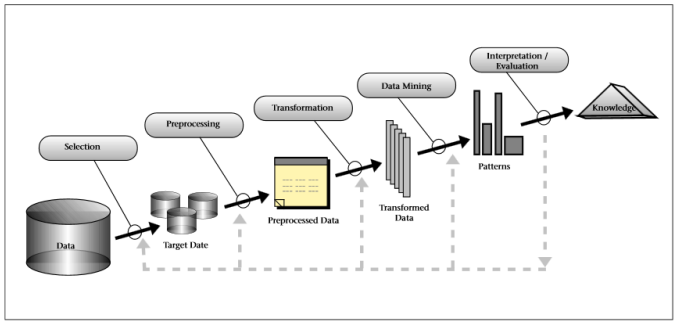

Figure 1. Prozessschritte im Data-Mining [1]

A. Prozessschritte des Text-Mining

Die einzelnen Prozessschritte im Text-Mining lassen sich aus dem in Abbildung 1 gezeigten Data-Mining-Prozess ableiten. Diese bestehen demnach aus den folgenden fünf Schritten:

- **Selection:** Im Zuge der *Selection* werden aus dem vorhandenen Datenbestand die zu verarbeitenden Textdokumente ausgewählt und gebündelt zur Verfügung gestellt.
- **Preprocessing:** Das *Preprocessing* beschreibt das Extrahieren von relevanten Informationen aus den vorhandenen Textdokumenten. Entscheidend ist hier die Art der Datenrepräsentation, anhand derer die gewünschten Mining-Methoden anwendbar sind. Zum Preprocessing zählt u. a. die Vorbereitung der Texte durch Normalisierung, Stemming (Reduzieren auf Wortstamm) und das eliminieren von *Stopwords*. Letztere sind Terme, die für die Performance der Analyse keinerlei Relevanz aufweisen [2]. Das Preprocessing wird im Abschnitt II-B im Detail betrachtet.
- **Transformation:** Hier gilt es, die Textdokumente in eine für die Mining-Algorithmen verwertbare Form zu transformieren. So wird nach dem *Bag-of-Words*-Ansatz aus dem Text ein n-dimensionaler Feature-Vektor erstellt. Dabei werden alle in den verwendeten Texten auftretenden, eindeutigen Terme in einem Feature-Raum zusammengefasst. Der Feature-Vektor kann zudem auf *Keywords* reduziert werden, die z. B. besonders häufig auftreten oder durch die sich aus Fachdomäne-spezifischen Metainformationen eine hohe Gewichtung ableiten lässt.
- **Data Mining:** Die verwendeten Mining-Algorithmen im Text-Mining dienen z. B. der Klassifizierung des Textes anhand einer Anzahl festgelegter Klassen oder dem Clustering von Datensätzen. Beide Anwendungsklassen setzten dabei voraus, dass ein Abstandsmaß in Form einer Distanzfunktion existiert, anhand derer sich Daten vergleichen lassen. Im Abschnitt II-C und II-D werden zwei für diesen Anwendungsfall interessante Algorithmen zur Clusteranalyse vorgestellt.
- **Interpretation:** Die resultierenden Ergebnisse des Text-Mining-Prozesses werden im Zuge der Interpretation zum Gewinn von Erkenntnissen verwertet.

Im Text-Mining ergeben sich besondere Anforderungen durch die Repräsentation der Daten als Feature-Vektor, dessen Dimension im *worst-case* dem Umfang eines Wörterbuches entsprechen kann. Dieses Wörterbuch umfasst alle bekannten Termen im Rahmen der Ontologie der verwendeten Domäne [3]. Im folgenden Abschnitt II-B wird diese Problematik thematisiert.

B. Preprocessiong und Transformation von Text

Bevor Mining-Algorithmen Textdaten zur Klassifizierung oder zum Clustering verwenden können, muss im Schritt des *Preprocessings* und der *Transformation* zunächst eine geeignete Repräsentation eines Textdokuments erstellt werden. Hier bietet sich das *Vector Space Model* (VSM) an [4]. Besonders verbreitet ist im VSM die Behandlung des Textes nach dem *Bag-of-Words*-Ansatz [5]. Demnach ist ein Dokument lediglich als eine Ansammlung von Wörtern zu betrachten, bei der die semantische Zusammenhänge der Wörter zueinander und die Grammatik des Textes keine Beachtung finden [2]. Das Modell betrachtet nur die Anzahl bzw. die Frequenz des Auftretens individueller Wörter in den zu analysierenden Texte. Die aus den Texten extrahierten Terme werden in diesem Modell als *Feature* bezeichnet. Der *Feature-Raum* umfasst damit alle in den Texten auftretenden Terme bzw. die domänenspezifischen Ontologie. Dieser Feature-Raum ist dabei in der Regel von sehr großer Dimension. Diese hohe Dimensionalität kann entsprechend negative Folgen auf die Performance der z. B. Clusteranalyse mit sich bringen [2]. Aus diesem Grund gilt die Reduzierung der Dimension der Feature-Vektoren zu den Herausforderungen des Preprocessings im Text-Mining [6][5].

Im Zuge des Preprocessings soll anhand der *Feature-Selection* die Anzahl der vorkommenden Terme reduziert werden [2]. Dazu müssen zunächst alle im Dokument enthaltene Wörter normalisiert und auf einen eindeutigen Wortstamm zurück geführt werden. Auch müssen die semantisch uninteressanten *Stopwords* [2] entfernt werden. Als nächstes kann die Frequenz des Auftretens der verbleibenden Wörter gezählt werden. Ein Textdokument lässt sich dann als *Feature-Vektor* im VSM betrachten, wobei die Dimension des Vektors durch die Anzahl der verschiedenen Wörter des Textes bestimmt wird [2].

Es existieren verschiedene Ansätze zur Reduzierung der hohen Dimension der Feature-Vektoren. Zunächst können einfach die am häufigsten auftretenden Features verwendet werden. Es zeigt sich dabei, dass eine Selektion von 10% der am häufigsten auftretenden Features die Performance der Clusteranalyse nicht beeinträchtigen [2].

Eine statistische Methode zur Reduktion der Dimension ist die *Principal Component Analysis* (PCA). Dabei betrachtet man die $n \times m$ Matrix der Dokumente und deren Terme und berechnet die k Eigenvektoren der PCA anhand einer $m \times m$ Kovarianzmatrix um schließlich den Feature-Raum auf die k-te Dimension zu veringern [6]. Ein weiteres Verfahren ist das z. B. in [7] angewendete *Latent Semantic Indexing* (LSI). LSI folgt bei der Reduzierung der Dimension des Feature-Raumes einem vergleichbaren Ansatz wie bei der PCA [6].

Neben der hohen Dimension von Feature-Vektoren im VSM ist auch das Fehlen von semantischen Informationen aus dem verwendeten Text ein bedeutender Nachteil. Durch semantische Zusammenhänge im Text kann z. B. die Gewichtung einzelner Terme im Feature-Vektor oder auch die Dimension des Feature-Vektors optimiert werden und so das Ergebnis von Klassifizierung oder Clustering verbessern. Daher bietet sich die Verwendung von Ontologien [8] und Thesauri [9] an, durch die sich die Methoden des Text-Mining gegenüber der einfachen Verwendung des VSM, basierend auf den Term-Frequenzen, optimieren lassen.

C. Clustering mittels k-Means

Der k-Means-Algorithmus ist ein ein einfaches und daher populäres Verfahren zur Clusteranalyse [10][2]. Der Begriff *k-Means* wurde bereits 1967 durch MacQueen geprägt [11]. Grundlegend partitioniert der Algorithmus eine Menge an Vektoren $\{x_1, x_2, ...x_n\}$ in eine vorgegebene Menge an Clustern $\{C_1, C_2, ...C_k\}$ [2]. Der Algorithmus basiert dabei auf lediglich zwei Vorgängen. Bei der *Initialization* werden aus der Menge an durch Vektoren repräsentierten Dokumenten zufällig

k Cluster-Zentren ausgewählt. Die übrigen Vektoren werden daraufhin dem anhand eines Abstandsmaßes bestimmten nächsten Zentrum zugeteilt. Die Qualität der somit zufällig bestimmten Cluster-Zentren und den sich darauf bildenden Cluster ist bereits kritisch für die Performance der Clusteranalyse mittels *k-Means* [10]. Im nächsten Schritt, der *Iteration*, werden die *Centroiden* als Mittelpunkte der bestehenden Cluster nach dem in Formel 1 beschriebenen Verfahren bestimmt [2]. Im Anschluss werden die einzelnen Vektoren jeweils dem nächsten *Centroiden* zugeteilt.

$$M_i = |C_i|^{-1} \sum_{x \in c_i} x \tag{1}$$

Die Iteration wird so oft wiederholt, bis die Abbruchbedingung zutrifft. Als Abbruchbedingung gilt dabei die Konvergenz, also sobald keine Veränderungen der Cluster mehr zu verzeichnen sind [2]. Der Abstand wird dabei in der Regel durch das Abstandsmaß nach *Euklid* berechnet. Die Euklidische Distanz ist in der Formel 2 beschrieben.

$$dist_{Euklid}(v, w) = \sqrt{\sum_i (v_i - w_i)^2} \tag{2}$$

Der *k-Means*-Algorithmus erreicht bei jeder Iteration eine Komplexität von $O(kn)$ bezüglich der Vergleiche zur Bestimmung der Ähnlichkeit [2] und benötigt meistens nur eine geringe Anzahl an Iterationen [10]. Die Nachteile von *k-Means* liegen dabei in der Abhängigkeit der Performance von der initialen und zufälligen Verteilung der Cluster. Ein einfacher Lösungsansatz ist hier das Festlegen initialer Zentren unter Verwendung von domänenspezifischen Kenntnissen [2]. So können z. B. zu der festgelegten Anzahl an Cluster durch den Anwender Leitartikel festgelegt werden, welche im Zuge der Clusteranalyse für die Erstellung der initialen Cluster verwendet werden können.

D. Clustering mittels SOMs

Selbstorganisierende Karten (SOM) oder auch *Kohonen-Netze* [12] nach dem finnischen Professor *Teuvo Kohonen*, sind eine Variante eines *künstlichen neuronalen Netzes* (KNN), welches zur Clusteranalyse von Daten verwendet werden kann. Ein KNN ist einem Modell des menschlichen Gehirns nachempfunden, wonach Neuronen die physiologischen Einheiten des Nervensystems bilden [13]. Die *Neuronentheorie* wurde dabei bereits u. a. durch *Santiago Ramon y Cajal* (1852-1934) begründet, welcher für seine Forschung den Nobelpreis für Physiologie oder Medizin erhielt [13]. Demnach wird anhand von KNN versucht, aus der Kenntnis der Forschung über das enorme Potential des menschlichen Gehirns bezüglich des Lernens und der Mustererkennung zu profitieren.
Das Prinzip hinter der Clusteranalyse mittels *Selbstorganisierender Karten* (SOM) basiert dabei auf einer Schicht aus geographisch angeordneten Neuronen, die in Regionen eingeteilt werden können. Diese Regionen werden dabei durch bestimmte Eingangsmuster unterschiedlich angeregt [6]. Mittels dieser vernetzten Schicht aus Neuronen kann ein viel-dimensionales Eingangsmuster auf einen Raum geringerer Dimension zu übertragen werden. Dabei bleiben die topologischen Zusammenhänge der viel-dimensionalen Eingangsmuster erhalten [14]. Das Modell der selbstorganisierenden Karten wurde in zahlreichen Publikationen aufgegriffen und durch zahlreiche Anpassungen optimiert [15]. Im Abschnitt II-D4 werden dazu Optimierungen einer SOM vorgestellt. Der Algorithmus der Selbstorganisierende Karten ist einer der am häufigsten verwendeten Algorithmen zur Datenanalyse [6].

1) Architektur von SOMs: Selbstorganisierende Karten bestehen im Allgemeinen aus zwei Schichten [10]:

- Eine aus künstlichen Neuronen bestehende **Eingabeschicht** nimmt die Muster der Feature-Vektoren an, die als Repräsentation der zu analysierenden Dokumente dienen.
- Die **Kartenschicht** ist eine geographische Anordnung aus Neuronen. Die Schicht bildet also eine Art Karte mit verschiedenen Regionen ab, die gleichzeitig als Ausgabeschicht dienen.

Die Neuronen der Eingabeschicht sind jeweils mit jedem einzelnen Neuron der Kartenschicht verbunden. Die Neuronen der Kartenschicht sind zudem untereinander implizit verbunden [10]. Die Verbindungen zwischen den Neuronen sind, wie in einem künstlichen neuronalen Netz üblich, mit einer Gewichtung versehen. Die Gewichtung einzelner Verbindungen ergibt sich aus dem geometrischen Abstand, den die Neuronen der Kartenschicht zueinander haben. In Abbildung 2 ist beispielhaft die Architektur einer selbstorganisierenden Karte abgebildet.
Zur Clusteranalyse von Textdokumenten bedarf es zunächst

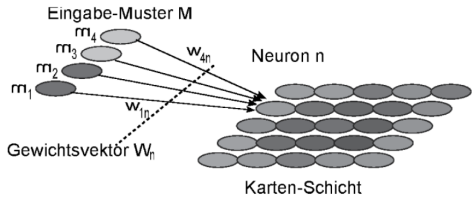

Figure 2. Allgemeine Architektur einer SOM [10]

einem Training. Die Neuronen der Karte werden zunächst zufällig verteilt [10] und erst nach einem hinreichenden Training bildet sich eine homogene Verteilung der Neuronen in der Kartenschicht, sodass Cluster erkennbar werden. Erst dann ist ein *Mapping* neuer Muster auf eine repräsentative Position der Karte möglich. Dabei wird durch das Eingangsmuster eines neuen Datensatzes ein einzelnes Neuron besonders stark angeregt [6]. Dieses Neuron und die direkten Nachbarn des Neurons in der Kartenschicht in einem bestimmten Radius bilden dann einen Cluster.
Der Nachteil einer SOM besteht in der Topologie, die zu Beginn festgelegt werden muss. Dabei lässt sich zu Beginn die optimale Topologie und damit die Größe der Kartenschicht nicht effektiv bestimmen [10]. Allerdings eignet sich bereits eine geringe Kartengröße von $10x10$ Neuronen um erste Ergebnisse zu erzielen [10]. Unter den Verbesserungen der SOM gibt es deshalb auch Ansätze zu einer in ihrer Größe dynamischen SOM, auf die im folgenden Abschnitt II-D4 eingegangen wird.

2) Trainieren einer SOM: Zu Beginn des Trainings ist nicht bekannt, welches bestimmte Muster ein Neuron am stärksten anregt. Die Gewichte der Neuronen in der Kartenschicht werden zufällig initialisiert [10]. Um die Clusteranalyse mittels der SOM zu verbessern werden nun in der Trainingsphase

zufällige Muster an die Kartenschicht angelegt [6]. Abbildung 2 zeigt dazu, wie für einen Vektor eines Eingabemusters M mit k Werten über die Menge der Verbindungsgewichte w_{in} zu einem Neuron n der Gewichtsvektor W_n gebildet werden kann [10]. Dieser Gewichtsvektor bestimmt die Aktivierung eines Neurons durch ein Eingabemuster. Der Vektor des Eingangsmusters und der Gewichtsvektor des betrachteten Neurons der Kartenschicht ermöglichen die Bestimmung eines Ähnlichkeitsmaßes. Die Ähnlichkeit lässt sich z. B. über den euklidischen Abstand $dist_{Euklid}(M, n)$ in Gleichung 2 bestimmen. Da es sich bei den zu vergleichenden Teilen um Vektoren handelt, bietet sich alternativ das *Cosinus-Ähnlichkeitsmaß* an, welches in Gleichung 3 beschrieben ist.

$$cos(M, W) = \frac{\sum_k m_i * w_{in}}{\sqrt{\sum_k m_i^2 * \sum_k w_{in}^2}} \tag{3}$$

Ist das Gewinner-Neuron bestimmt, welches den geringsten Abstand zu dem Eingangsmuster aufweist, kann im Zuge der Trainingsphase die Kartenschicht optimiert werden. Das Ziel ist hier, die Neuronen der Kartenschicht auf der Karte besser zu verteilen. Um dies zu erlangen, werden die Gewichte des Gewinner-Neurons zu der Eingangsschicht angepasst und damit auch die Position des Neurons auf der Karten-Schicht entsprechend verändert [6]. Damit wird also das entsprechende Gewinner-Neuron dem Eingangsmuster ähnlicher [10]. Neben der Anpassung des Gewinner-Neurons werden auch die Nachbarn des Gewinner-Neurons angepasst, sofern sie sich in einem zuvor bestimmten Radius zu dem Gewinner-Neuron befinden. Die Anpassung der Gewichte ist in der Formel 4 beschrieben. Durch λ wird dabei der Lernparameter beschrieben, der die Veränderungen im Lernprozess beeinflusst. Der Parameter h dient dazu der Beeinflussung der Gewichtsänderung in Abhängigkeit zu dem Abstand des betrachteten Neurons j zu dem zuvor bestimmten Gewinner-Neuron. Durch r wird der gültige Radius zur Anpassung der Nachbarn des Gewinner-Neurons bestimmt [10].

$$w'_{ij} = \begin{cases} w_{ij} * \lambda * h_{jz}) * (m_i - w_{ij}) & \text{falls } dist(j, z) \leq r \\ w_{ij} & \text{sonst} \end{cases} \tag{4}$$

Der vom Abstand abhängige Parameter zur Gewichtsänderung h lässt sich nach Formel 5 berechnen [10].

$$h_{jz} = e^{-\frac{dist(j,z)^2}{2r^2}} \tag{5}$$

Zu Beginn der Lernphase wird also zunächst die Kartenschicht und damit die Neuronen und ihre gewichteten Verbindungen zufällig initialisiert. Im Anschluss werden in einzelnen Iterationen alle existierenden Eingangsmuster in einer zufälligen Reihenfolge an das Netz angelegt und das entsprechende Gewinner-Neuron der Kartenschicht bestimmt. Gemäß Formel 4 und 5 können dann die Gewichte und damit auch die Position des Gewinner-Neurons und der benachbarten Neuronen angepasst werden. Damit der Lernprozess terminieren kann, werden nach jedem Zyklus der Lernphase Lernparameter λ und Radius r verkleinert. Damit kann der Lernprozess mit $\lambda = 0$ oder $h = 0$ terminieren [10].

3) Mapping von Textdaten: Nach einem hinreichenden Lernprozess sind die Neuronen der Kartenschicht einer SOM auf bestimmte Eingangsmuster trainiert. Ähnliche Eingabemuster werden dann in der Kartenschicht eines der benachbarten Neuronen anregen [6]. Bei der Verarbeitung einer Menge an Textdokumenten werden also ähnliche Dokumente auf benachbarten Regionen der Kartenschicht abgebildet und es lassen sich anhand einer geeigneten Visualisierung Cluster erkennen. Abbildung 4 zeigt eine SOM mit einem zweidimensionalen Eingaberaum, in dem sich Muster als *x*-*y*-Koordinaten wiederfinden lassen [10]. Die gezeigte selbstorganisierende Karte weißt dabei in ihrer Struktur der Neuronen eine Heterogenität auf. Durch intensiveres Training lässt sich die Topologie der Neuronen verbessern, sodass in einer Abbildung wie in 4 ein gleichmäßiges Gitternetz der Neuronen der Kartenschicht zu erkennen ist.

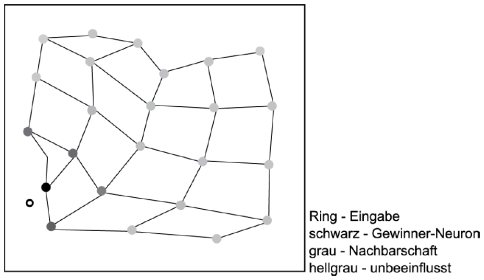

Figure 3. SOM im zweidimensionalen Eingaberaum [10]

4) Optimierungen einer SOM: Zu dem Modell der selbstorganisierenden Karten existieren zahlreiche Optimierungen, die Probleme bei der Verwendung von selbstorganisierenden Karten zur Clusteranalyse lösen sollen. Ein solches Problem liegt z. B. darin, dass die Topologie der Kartenschicht der SOM vorgegeben werden muss [6]. Es existieren dazu verschiedene Ansätze, die das Modell der SOM hinsichtlich der dynamischen Erweiterung der Neuronen-Topologie modifizieren. Eine Modifikation ist das in [6] vorgeschlagene *V-SOM*-Verfahren. Bei dem Verfahren wird eine ringförmige Topologie der Ausgangsschicht verwendet, die das dynamische Hinzufügen neuer Neuronen vereinfacht [6]. Eine solche ringförmige Topologie ist in Abbildung 3 dargestellt. Dazu wird anhand der ringförmigen Topologie das Hinzufügen eines neuen Neurons in einer V-SOM veranschaulicht. Die Identifikationsnummer eines bestimmten Sektors der Ring-Topologie wird dabei durch einen beliebigen Integer-Wert dargestellt. Im Modell der V-SOM sollen zudem die Neuronen der Kartenschicht besser ausgenutzt und damit ungenutzte Neuronen verhindert werden [6]. Ein weiterer Ansatz ist die *Topic-Orientated*

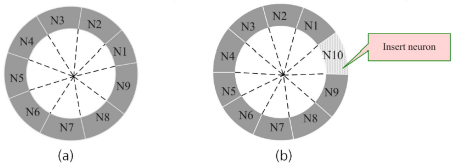

Figure 4. V-SOM Ring-Topologie [6]

Self-Organizing Map (TOSOM), welche in [14] beschrieben ist. Im Gegensatz zu einer SOM werden im Lernprozess der TSOM zu den einzelnen Neuronen der Karten-Schicht *Topics* ermittelt, die ein Dokument repräsentieren. Durch diese Praxis werden zusätzliche Informationen erlangt, anhand derer im Zuge der Clusteranalyse entschieden werden kann, ob die

Kartenschicht der TSOM seitlich oder hierarchisch erweitert werden soll [14]. Die Karte lässt sich somit dynamisch an die zu verarbeitenden Textdokumente anpassen und es kann das Ergebnis der Clusteranalyse verbessert werden.

III. Tools im Text-Mining

Im Bereich des Data-Mining existieren zahlreiche Tools und Frameworks, die den Prozess des Data-Mining in großen Teilen implementiert haben und so dem Anwender eine unkomplizierte Plattform für anwendungsspezifische Mining-Aufgaben bietet. Für das Text-Mining existiert allerdings hinsichtlich des *Preprocessing* keine bereits vollständigen Implementierung. Dies ist darauf zurück zu führen, dass sich das Preprocessing in den anwendungsspezifischen Domänen stark unterscheidet.
Im folgenden Abschnitt III-A wird das Tool *RapidMiner* vorgestellt, welches einen einfachen Zugang zu Mining-Prozessen liefert. Im Gegensatz dazu beschäftigt sich Abschnitt III-B mit Frameworks, die auf Basis von Apache Hadoop zum Text-Mining verwendet werden können.

A. Rapidminer

RapidMiner ist eine an der Universität Dortmund entwickelte Umgebung für verschiedene Anwendungen aus dem Bereich Data-Mining. Dazu implementiert RapidMiner eine große Anzahl an Verfahren, die vergleichbar zu *Mathlab Simulink* nach einen Baustein-Ansatz zusammengeschaltet werden können. Dadurch können z. B. Verfahren zum Text-Mining oder zur *Sentiment Analyse* realisiert werden. Durch die Möglichkeit, komplexe Data-Mining-Verfahren über eine graphische Oberfläche zusammenzustellen, ist RapidMiner besonders zum Einstieg und für Forschung interessant. RapidMiner ist zudem in *Java* geschrieben und ermöglicht das Hinzufügen von durch den Anwender erstellten Modulen. Anhand dieser lassen sich etwa eigene Implementierungen von Algorithmen testen und in einen Data-Mining-Prozess integrieren.

B. Apache Hadoop zum Text-Mining

Apache Hadoop ist der de-facto Standard im Bereich der verteilten Verarbeitung von großen Datenmengen. Das liegt zum einen an der hochgradigen Skalierbarkeit eines Hadoop-Systems und zum anderen an der modularen Erweiterbarkeit von Hadoop als Plattform zur Datenverarbeitung. Im Gegensatz zu dem im vorangehenden Abschnitt III-A beschriebenen Tool *RapidMiner* bietet Apache Hadoop zunächst lediglich eine Plattform für das verteilte Verarbeiten von großen Datenmengen. Hadoop besteht dabei grundsätzlich aus zwei Komponenten. Das *Hadoop Distributed Filesystem* (HDFS) ist ein auf Googles *BigTable* aufbauendes verteiltes Filesystem [16]. Neben dem HDFS bietet *Hadoop YARN* einen Scheduler und Ressourcen-Manager, mit dem die verteilten Anwendungen auf dem Hadoop-System kommunizieren [16]. Neben den im Hadoop-Ökosystem vorhandenen Frameworks zur Datenverarbeitung können auch eigene verteilte Anwendungen auf Hadoop realisiert werden. Diese müssen dazu lediglich eine *ApplicationMaster*-Komponente implementieren, die über den Ressourcen-Manager YARN Ressourcen des Hadoop-Clusters anfordert [16]. Auch können Daten aus dem HDFS abgerufen und darin gespeichert werden.
Ein weit verbreitetes Tool zum Data-Mining, welches auf Hadoop aufbaut, ist *Apache Mahout*. Mahout bietet ähnlich dem zuvor vorgestellten RapidMiner bereits Implementierungen von Verfahren zum Text-Mining. Im Gegensatz zum RapidMinder können dabei die Mining-Verfahren vollständig verteilt realisiert werden. Dies mach Hadoop besonders für Produktiv-Systeme interessant.

IV. Aktuelle Arbeiten und Projekt-Ausblick

Text-Mining wird bereits in verschiedenen Anwendungsszenarien erfolgreich eingesetzt. Dabei unterscheiden sich die Ergebnisse allerdings von einem Anwendungsfall zum anderen. Im Bereich des Data-Mining gibt es zahlreiche effiziente und hinreichend getestete Implementierungen von Algorithmen zur Clusteranalyse. Das in Abschnitt II-D vorgestellte Verfahren zur Clusteranalyse anhand von SOMs findet in vielen Beispielen Anwendung [6] und es existieren zahlreiche optimierte Ansätze hinsichtlich der Fähigkeiten zum Clustering und des Lernprozesses. Im Gegensatz dazu existiert bezüglich der Vorverarbeitung von Texten zur Verarbeitung durch die gängigen Data-Mining-Algorithmen kein allgemeiner Konsens. Im folgenden Abschnitt IV-A werden hierzu einige aktuelle Arbeiten herangezogen, die sich mit einem vergleichbaren Anwendungsfall beschäftigen.

A. Aktuelle Arbeiten

Zunächst stellt die Anwendung von Text-Mining zur automatischen Analyse von politischen Text-Inhalten [17] heraus, dass Text-Mining-Prozesse aufgrund der Komplexität von Sprache derzeit nur als Unterstützung zum manuellen Textverständnis zu sehen sind. Ein ähnlicher Ansatz wird in [18] verfolgt, indem durch die Webanwendung *PubTator* ein Portal geschaffen wurde, welches anhand von Text-Mining das manuelle Erfassen von medizinischen Veröffentlichungen des Portals *PubMed* vereinfachen soll.
Auch gibt es Forschung zur Anwendung von Text-Mining zur Erkennung von *Opinion Spam* in [19], die zum Ergebnis kommt, dass unter Verwendung von domänenspezifischen Kenntnissen der Linguistik und Psychologie Ergebnisse erzielt werden können, welche die händische Erkennung von *Opinion Spam* durch Menschen übertreffen.
In [5] wird das Text-Mining dazu verwendet, um anhand von Nachrichten der Wirtschaft die Entwicklungen an der Börse voraus zu sagen. Dabei wird ein guter Überblick der derzeitig verwendeten Methoden gegeben. So werden beispielsweise Methoden zur Reduzierung des Feature-Raumes aus vorangehenden Arbeiten abgehandelt.
Ein besonders für diese Arbeit interessanter Ansatz ist in [20] zu finden. Hier wird das Text-Mining-Verfahren durch Ausnutzung der domänenspezifischen Ontologien und unter Verwendung von *Natural Language Processing* verbessert.
Eine weitere interessante Arbeit [21] beschreibt die Implementierung des *Stanford Natural Language Processing Tool Set* zur Verwendung in Text-Mining-Prozessen. Anhand dieses Tool-Sets lassen sich Verfahren wie z. B. das Segmentieren von Text oder das *part-of-speech tagging* auf die zu analysierenden Dokumente anwenden. Aus diesen Analysen lassen sich so möglicherweise wichtige Informationen gewinnen, die zur *Feature Selection* verwendet werden können und das Ergebnis der Clusteranalyse verbessern.

B. Schnittstellen zu anderen Arbeiten

Schnittstellen dieser Arbeit gibt es zu den Ausführungen von *Marcel Schöneberg*, der sich in [22] unter der Betreuung von *Prof. Dr. Kai von Luck* bereits mit dem Thema auseinandergesetzt hat. Unter Verwendung einer einfachen Clusteranalyse werden dabei die Dokumente aus der Datenbasis des *Eurozine*-Netzwerkes verarbeitet. Besonderer Fokus liegt auf der optimalen Gewichtung der Features eines Dokuments, um im Zuge der Clusteranalyse ein aussagekräftiges Ergebnis zu erzielen. In [22] wird zudem *RapidMiner* zur Realisierung des Text-Mining verwendet.

C. Ausblick

Auf Basis der derzeitigen Erkenntnisse, welche u. a. in den im vorangehenden Abschnitt IV-A beschriebenen Ausarbeitungen vorgestellt werden, soll im Verlauf des Master-Projektes ein Text-Mining-Prozess zum automatisierten Erstellen von journalistischen Dossiers realisiert werden. Abbildung 5 zeigt in Form eines System-Diagramms einen Ausblick zu der im Projekt geplanten Arbeit. Zunächst gilt es natürlich den Text-Mining-Prozess in seiner Gesamtheit zu implementieren. Da allerdings die Methoden zum Data-Mining hinreichend implementiert sind und derzeit bereits breit verwendet werden, gilt dem *Preprocessing* besonderer Fokus. Hier müssen zu der gegebenen Fachdomäne die vorhandenen Kenntnisse dazu genutzt werden, domänenspezifische Ontologien zu definieren und ggf. hierarchische Thesauri zu bilden. Diese sollen dann im *Preprocessing* verwendet werden, um den Feature-Raum der zu analysierenden Dokumente so zu reduzieren bzw. die Ausprägungen der einzelnen Dokumente derart zu gewichten, dass im Zuge der Clusteranalyse brauchbare Ergebnisse erzielt werden können. Demnach sollen anhand des gegebenen Dokumentenkorpus durch eine Clusteranalyse aussagekräftige Cluster erzeugt werden.

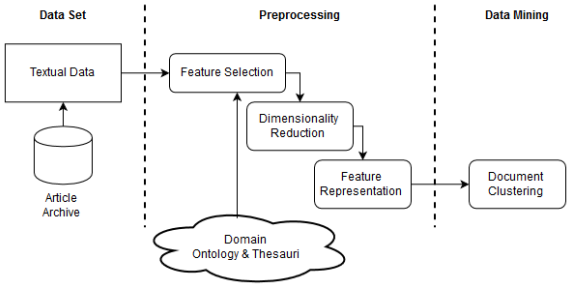

Figure 5. System-Diagramm als Projekt-Ausblick (vgl. [5])

Ist diese Anforderung hinreichend erfüllt, kann das Gesamtsystem durch ein Modul erweitert werden, welches einem Anwender die Erstellung von journalistischen Dossiers erleichtern soll. Abbildung 6 zeigt dazu die vorgesehenen Prozessschritte. Der Anwender soll über eine Webanwendung auf das System zur Dossier-Erstellung zugreifen können. Dabei wird der Fokus darauf liegen, dem Anwender in Form eines Vorschlagsystems vergleichbar zu *PubTator* [18] eine Hilfestellung zur Erstellung eines Dossiers zu bieten. Der Anwender soll dazu beispielsweise ein Dokument als Leitartikel vorgeben. Auch kann der Dokumentenkorpus durch den Anwender festgelegt werden. Durch ein geeignetes Text-Mining-Verfahren werden dann die zur Verfügung stehenden Artikel anhand einer Clusteranalyse

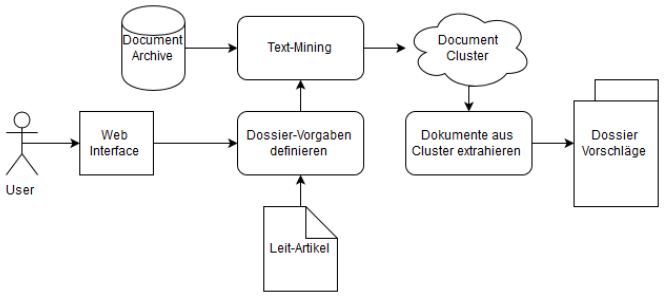

Figure 6. Gesamt-System zur Dossier-Erstellung

verarbeitet. Daraufhin kann der Leitartikel in dem Ergebnis der Clusteranalyse lokalisiert werden und durch ein vorgegebenes Mindestmaß an Ähnlichkeit bzw. eines Abstandes die den Leitartikel umgebenden Dokumente extrahiert werden. Diese Dokumente werden dann dem Anwender zur Erstellung des Dossiers vorgeschlagen. Damit soll die Produktivität des Anwenders zur Dossier-Erstellung maßgeblich verstärkt werden. Dieser Ansatz folgt dabei der in [17] aufgestellten These, dass Text-Mining-Verfahren besonders zur Unterstützung des Menschen in entsprechenden Prozessen eingesetzt werden können. Vom technischen Standpunkt her ist also ein Gesamt-System zu entwickeln, welches die einzelnen Funktionalitäten erfüllt oder bereits bestehende Software für diese Funktionalitäten integriert. Es existieren dazu verschiedene Tools, welche zum Text-Mining verwendet werden können. Das im Abschnitt III-A vorgestellte Tool *Rapidminer* ist dabei eine modulare Gesamt-Lösung, bei der einzelne Prozessschritte im Data-Mining hintereinander gefügt werden können. Einzelne Bausteine sind dabei entweder bereits implementiert oder können manuell in Java ausgeschrieben werden. Dies ermöglicht einen einfachen Ansatz zur Realisierung der Text-Mining-Komponente des angedachten Gesamtsystems.

Im Gegensatz dazu bietet das im Abschnitt III-B beschriebenen Framework *Apache Hadoop* lediglich die Grundlage für verteiltes Speichern und Verarbeiten von Daten. Allerdings lassen sich anhand von auf *Hadoop YARN* aufbauende Anwendungen wie z. B. *Hadoop Mahout* Data-Mining-Aufgaben verteilt lösen. Dabei werden die Daten im *HDFS*, dem verteilten Filesystem von Hadoop gespeichert und können von *Mahout* verwendet werden. *Mahout* implementiert bereits Clustering-Algorithmen wie z. B. das in Abschnitt II-C beschriebene k-Means-Clustering und Methoden zur Reduktion der Dimension eines Feature-Raumes[1]. Ein Preprocessing der Dokumenten kann als selbstständige Software für Hadoop YARN entwickelt werden. In diesem Umfeld hat der Autor bereits im Zuge der Bachelorarbeit [23] Erfahrungen gesammelt. Der Vorteil von Hadoop liegt klar in der Skalierbarkeit und dem parallelen Verarbeiten der Daten über das verteilte Filesystem. Auch existiert ein besonders lebendiges Ökosystem von Frameworks, die im Umfeld des Data-Mining angesiedelt sind.

Aus den genannten Gründen bietet sich ein Tool wie z. B. *Rapidminer* für die Entwicklung eines ersten Prototyps eines Text-Mining-Systems an. Auf längere Sicht erscheint es allerdings als sinnvoll, eine Realisierung des Gesamtsystems auf *Apache Hadoop* als Plattform zu entwickeln.

[1] Quelle: http://mahout.apache.org/users/basics/algorithms.html

V. Zusammenfassung

In dieser Ausarbeitung wurde ein Überblick zu den Prozessen des Text-Mining hinsichtlich der Entwicklung eines Systems für Vorschläge zu journalistischen Dossiers gegeben. Zur Realisierung eines solchen Systems soll ein Text-Mining-Verfahren implementiert werden, welches zur Clusteranalyse der zur Verfügung stehenden Dokumente verwendet werden kann. Anhand der Clusteranalyse lassen sich so zu einem vorgegebenen Leitartikel ähnliche Texte finden, die dem Anwender zur Erstellung von Dossiers vorgeschlagen werden können.

Aus aktuellen Arbeiten lässt sich erkennen, dass die Methoden zur Clusteranalyse im Text-Mining derzeit breite Anwendung finden und hinreichend optimiert sind. Es lässt sich allerdings auch ableiten, dass es für die Vorverarbeitung von Text nach aktuellem Stand kein optimales Verfahren gibt. Die Herausforderung der kommenden Arbeiten liegt also in der geschickten Vorverarbeitung unter Verwendung von beispielsweise domänenspezifischen Ontologien oder Methoden aus dem *Natural Language Processing*.

Im Zuge der Projekt-Veranstaltung soll ein Text-Mining-System realisiert werden, welches die Clusteranalyse von Textdokumenten ermöglicht. Dabei soll besonderer Fokus auf die Vorverarbeitung der Textdokumente gelegt werden und so ein aussagekräftiges Ergebnis erzielt werden. Ist dieses Ziel erfüllt, wird über eine Webanwendung ein Vorschlags-System entwickelt, welches zur Unterstützung der Erstellung von journalistischen Dossiers dienen kann.

References

[1] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "Advances in knowledge discovery and data mining," U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996, ch. From Data Mining to Knowledge Discovery: An Overview, pp. 1–34. [Online]. Available: http://dl.acm.org/citation.cfm?id=257938.257942

[2] R. Feldman and J. Sanger, The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press, 2007. [Online]. Available: https://books.google.de/books?id=U3EA_zX3ZwEC

[3] R. Navigli and P. Velardi, "Learning domain ontologies from document warehouses and dedicated web sites," Comput. Linguist., vol. 30, no. 2, Jun. 2004, pp. 151–179. [Online]. Available: http://www.dsi.uniroma1.it/~{}velardi/CL.pdf

[4] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," Commun. ACM, vol. 18, no. 11, Nov. 1975, pp. 613–620. [Online]. Available: http://doi.acm.org/10.1145/361219.361220

[5] A. K. Nassirtoussi, S. Aghabozorgi, T. Y. Wah, and D. C. L. Ngo, "Text mining for market prediction: A systematic review," Expert Systems with Applications, vol. 41, no. 16, 2014, pp. 7653–7670.

[6] M. L. Yuan-Chao Liu and X.-L. Wang, "Applications of self-organizing maps," M. Johnsson, Ed. InTech, 2012, ch. Application of Self-Organizing Maps in Text Clustering: A Review. [Online]. Available: http://www.intechopen.com/books/applications-of-self-organizing-maps/application-of-self-organizing-maps-in-text-clustering-a-review

[7] K. Krishnamurthi, R. K. Sudi, V. R. Panuganti, and V. V. Bulusu, "An empirical evaluation of dimensionality reduction using latent semantic analysis on hindi text," in Proceedings of the 2013 International Conference on Asian Language Processing, ser. IALP '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 21–24. [Online]. Available: http://dx.doi.org/10.1109/IALP.2013.11

[8] S. Staab and R. Studer, Handbook on Ontologies, 2nd ed. Springer Publishing Company, Incorporated, 2009.

[9] K. Sparck Jones and P. Willett, Eds., Readings in Information Retrieval. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997.

[10] J. Cleve and U. Lämmel, Data Mining. Gruyter, de Oldenbourg, 2014. [Online]. Available: http://books.google.de/books?id=4i2nngEACAAJ

[11] J. MacQueen et al., "Some methods for classification and analysis of multivariate observations," in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, vol. 1, no. 14. Oakland, CA, USA., 1967, pp. 281–297.

[12] T. Kohonen, M. R. Schroeder, and T. S. Huang, Eds., Self-Organizing Maps, 3rd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001.

[13] M. Ronald S. Fishman, "The nobel prize of 1906," Archives of Ophtalmology, vol. 125, no. 5, May 2011, pp. 690–694.

[14] H.-C. Yang, C.-H. Lee, and K.-L. Ke, "Tosom: A topic-oriented self-organizing map for text organization," vol. 4, no. 5, 2010, pp. 865 – 869. [Online]. Available: http://waset.org/Publications?p=41

[15] S. Kaski, J. Kangas, and T. Kohonen, "Bibliography of self-organizing map (som) papers: 1981–1997," Neural computing surveys, vol. 1, no. 3&4, 1998, pp. 1–176.

[16] A. C. Murthy, V. K. Vavilapalli, D. Eadline, J. Niemiec, and J. Markham, Apache Hadoop YARN: Moving Beyond MapReduce and Batch Processing with Apache Hadoop 2. Pearson Education, 2013.

[17] J. Grimmer and B. M. Stewart, "Text as data: The promise and pitfalls of automatic content analysis methods for political texts," Political Analysis, vol. 21, 2013, pp. 267–297.

[18] C.-H. Wei, H.-Y. Kao, and Z. Lu, "Text mining tools for assisting literature curation," in Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, ser. BCB '14. New York, NY, USA: ACM, 2014, pp. 590–591. [Online]. Available: http://doi.acm.org/10.1145/2649387.2660786

[19] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, "Finding deceptive opinion spam by any stretch of the imagination," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011, pp. 309–319.

[20] A. Cheptsov, A. Tenschert, P. Schmidt, B. Glimm, M. Matthesius, and T. Liebig, "Introducing a new scalable data-as-a-service cloud platform for enriching traditional text mining techniques by integrating ontology modelling and natural language processing," in Web Information Systems Engineering–WISE 2013 Workshops. Springer, 2014, pp. 62–74.

[21] D. C. Comeau, H. Liu, R. I. Doğan, and W. J. Wilbur, "Natural language processing pipelines to annotate bioc collections with an application to the ncbi disease corpus," Database, vol. 2014, 2014, p. bau056.

[22] M. Schhöneberg, "Automatisierte Erstellung von Pressedossiers durch Textmining: Kontextualierung im journalistischen Umfeld," Arbeitspapier, HAW Hamburg, 2015.

[23] J. P. Assendorp, "Skalierbare Prozessierung von Satellitendaten," Bachelorarbeit, HAW Hamburg, 2014.