SEMINAR

FÜR WIRTSCHAFTSINFORMATIK

UND SYSTEMENTWICKLUNG

Prof. Dr. Werner Mellis

Hauptseminar Wirtschaftsinformatik

im Sommersemester 2015

Thema 30:

Entwicklung einer Anwendung zur automatisierten Inhaltsanalyse von Abstracts mittels Centering Resonance Analysis

vorgelegt von:

Kreuzer, Adrian

Inhaltsverzeichnis

Abkürzungsverzeichnis

CRA	Centering Resonance Analysis
NTA	Network Test Analysis
APSP	All-Pair-Shortest-Path
SSSP	Single-Source-Shortest-Path

1. Einleitung

Eine systematische Liteaturanalyse ist ein essentieller Bestandteil eines jeden wissenschaftlichen Projekts.[1] Sie schafft eine Wissensgrundlage und zeigt Bereiche auf, die bereits in großer Zahl behandelt wurden und solche, die weitere Forschung benötigen. Eine Analyse kann bei großer Anzahl verfügbarer Literatur, die stetig wachsen kann, sehr komplex und zeitaufwendig sein. Um beurteilen zu können, ob ein Text für den Leser relevant ist eignet sich das Lesen des jeweiligen Abstracts, da diese einen kurzen Abriss eines Textes wiedergeben.[2] Vorangehende Forschungen beschreiben verschiedene Vorgehensweisen, um eine systematische Literaturanalyse effizient durchzuführen. Dabei werden zum Beispiel Konzepte oder Suchbegriffe systematisch identifiziert und abgearbeitet.[3] Ist dem Autor einer Abhandlung ein Themengebiet unbekannt, so erfordert dies jedoch zuvor das Lesen von Texten, um diese Konzepte und Begriffe zu identifizieren. Gute Einstiegsliteratur ist unter Umständen nicht bekannt. Dies führt zu der Frage nach der Möglichkeit der Beschleunigung der Identifikation von Konzepten.

Mit Centering Resonance Analysis (CRA) existiert ein Textanalyse Algorithmus, der ein Netzwerk der Begriffe eines Textes bilden und darin die wichtigsten Begriffe identifizieren kann.[4] Forschungen und vorherige Anwendungen von CRA auf verschiedene Texte haben gezeigt, dass er dabei bessere Ergebnisse als andere Textanalyse Algorithmen liefert. Wie zuvor bereits erwähnt, geben Abstracts einen kurzen Überblick über einen Text. Darüber hinaus bieten einige wissenschaftliche Datenbanken, wie Ebscohost oder Proquest, die Möglichkeit alle Abstracts von Texten, die für einen Suchbegriff gefunden wurden, schnell in gängige Formate Dateien zu exportieren. Die Anwendung von CRA auf mehrere Abstracts könnte also die wichtigsten Begriffe dieser Abstracts beziehungsweise der zugehörigen Texte in liefern. Bisher wurde jedoch nicht untersucht, ob dies zu brauchbaren Ergebnissen führt bzw. ob dieses Vorgehen praxistauglich ist.

---

[1] Vgl. zu diesem Absatz Webster, Watson (2002), S. xiii sowie Bandara, Miskon, Fielt (2011), S. 1.

[2] Vgl. Michie, Williams (2003), S. 3–4.

[3] Vgl. Wolfswinkel, Furtmueller, Wilderom (2013). S. 45-47.

[4] Vgl. zu diesen und dem folgenden Satz Corman u. a. (2002), S. 158–200.

Darüber hinaus kann CRA die Resonanz verschiedener Texte vergleichen.

Ziel dieser Arbeit ist es zu untersuchen, ob durch die Anwendung von Centering Resonance Analysis auf Abstracts, die Litereraturrecherche unterstützt werden kann. Hierzu werden Anforderungen an eine Software für Textanalysen auf Basis von CRA erhoben. Anschließend wird der CRA-Algorithmus zunächst implementiert und anschließend seine Ergebnisse für Abstracts evaluiert. Neben den durch CRA ausgegebenen Begriffen, wird ebenfalls die Laufzeit bewertet.

2. Grundlagen

Die Basis einer wissenschaftlichen Arbeit ist die Literaturanalyse, auch Literature Review genannt. Laut Webster und Watson (2002) ist die Literarturanalyse ein fundamentales Mittel eines jeden akademischen Projekts.[5]

Die Einarbeitung in ein Themengebiet sowie die Basis für das Wissen über ein Themengebiet erreicht man, so Webster und Watson, durch eine Literaturanalyse.

Bandara et al. beschreiben die Relevanz eines systematischen Ansatzes für die Literaturanalyse.[6] Dabei gehen sie auf die Suche nach Schlagwörtern in Titel, Abstracts und Schlüsselwörtern eines Textes ein. Ein Abstract ist ein Abriss einer wissenschaftlichen Arbeit, der sowohl wertungs- als auch interpretationsfrei ist. Michie und Williams schreiben, dass Abstracts für die Identifikation von relevanter Literatur eine wichtige Quelle sind.[7] Eine Abstract Analyse, also das systematische analysieren von Abstracts, erhöht das Wissen über ein Themengebiet. Dadurch wird es dem Verfasser wissenschaftlicher Arbeiten möglich, neben der Erweiterung des eigenen Wissens über ein Themengebiet, weitere Suchbegriffe zu filtern, die ein breiteres Spektrum an relevanter Literatur darbieten. Die Wahl der richtigen Suchbegriffe und damit verbunden die Genauigkeit innerhalb eines systematischen Literatursuchprozesses

---

[5] Vgl. Wesbter, Watson (2002), S. xiii

[6] Vgl. Bandara, Miskon, Fielt (2011), S. 7

[7] Vgl. Michie, Willimas (2003), S. 3

bildet die Basis für eine erfolgreiche Literaturanalyse und erhöht den Erfolg der wissenschaftlichen Arbeit.[8]

Dieses Vorgehen kann durch Algorithmen unterstützt werden. Mit Hilfe dieser Textanalyse-Algorithmen werden Themen und zentrale Begriffe extrahiert. Art der Analyse, sowie Vorgehen und Ergebnisse einzelner Verfahren werden ausführlich in Kapitel 4 vorgestellt. Für die Suche nach relevanter Literatur, deren Ergebnisse durch den Verfasser einer wissenschaftlichen Arbeit auf Güte analysiert werden müssen, bieten diese Algorithmen die Möglichkeit auf Generierung weiterer Suchbegriffe.

2.1 Grundlagen der Netzwerk-Textanalyse

Mithilfe der sogenannten Netzwerktextanalyse, im Folgenden NTA genannt, können relevante Begriffe aus Texten extrahiert werden.[9] Der Begriff Netzwerk-Textanalyse umfasst hierbei für unsere Zwecke eine Menge von computerunterstützten Lösungen zur Erstellung eines Netzwerkes von Konzepten aus Texten.[8] Diese Analyse dient sowohl der grafischen Darstellung der Inhalte des jeweiligen Textes als auch der Extraktion und Ermittlung wichtiger Elemente. Hunter stellt hierbei mehrere mögliche Varianten der Analyse vor.[10] Diese verfolgen jeweils unterschiedliche Herangehensweisen, um ein Netzwerk (Graph) aus einem Text zu erstellen, dem Inhalt Bedeutung zuzuweisen und diesen aufbereitet darzustellen.

2.2 Netzwerk-Textanalyse mithilfe von Centering Resonance Analysis

Das für die bereits erwähnte Implementierung genutzte Verfahren ist die Centering Resonance Analysis (CRA). Sie gehört zu den Verfahren der Netzwerk-Textanalyse und identifiziert wichtige Worte eines Textes oder einer Aussage. Diese repräsentiert sie in Form von Netzwerken, welche miteinander verglichen werden können.[11] Letzteres geschieht durch die Eigenschaft der sogenannten Resonanz, die dazu dient, entstandenen Netzwerke miteinander zu vergleichen.[12] Durch die Darstellung in Form

---

[8] Vgl. vom Brocke u.a. (2009), S.4

[9] Vgl. Hunter (2014), S.350.

[10] Vgl. Hunter (2014), S.351.

[11] Vgl. Corman u.a. (2002), S.189.

[12] Vgl. Corman u.a. (2002), S.189.

eines solchen Netzwerks kann das Lesen eines oder mehrerer Texte substituiert werden.[13] CRA basiert, ähnlich wie andere NTA-Verfahren, in seinen Grundlagen auf vier Schritten: Der Selektion von Begriffen, die für die Erkennung der Bedeutung relevant erscheinen, der Vereinheitlichung oder Konzeptualisierung dieser Begriffe, der Verbindung dieser Begriffe und der Art, wie diesen Begriffen Relevanz zugeordnet wird.[14] Im Folgenden werden die jeweiligen Schritte im Bezug auf CRA erläutert und weitere Möglichkeiten der Extraktion von Bedeutung der einzelnen Begriffe vorgestellt. CRA nutzt hierbei das linguistische Konzept der der Zentralität, um wichtige Wörter oder Begriffe in Aussagen zu identifizieren und miteinander zu verbinden.[15] Hunter hat unterschiedliche Gruppen von Ansätzen der NTA miteinander verglichen und deren Vorgehensweisen in den jeweiligen Phasen beschrieben.[16] Zu diesen gehören sogenannte Semantische Netzwerke, Semantisches mapping, NTA in Sozialwissenschaften und CRA. Da sich die in dieser Arbeit beschriebene Implementierung auf die Anwendung des CRA in englischer Sprache konzentriert, werden in dieser Arbeit Beispiele in englischer Sprache verwendet.

2.2.1 Selektion

Die Selektion ist der erste Schritt der Analyse. Hierbei wird ein Text auf die für die Analyse relevanten Elemente reduziert. NTA-Ansätze wie Semantische Netzwerke oder Semantische Maps führen diese Selektion auf Basis der Häufigkeit des Auftretens von Wörtern durch, die für die jeweilige Methode relevant erscheinen und lassen hierbei grundsätzlich bestimmte Wortarten aus. Diese Wortarten sind zum Artikel wie „a, an, the“ als auch Pronomen, Präpositionen, Konjunktionen, Verben, Akronyme und Namen.[13] Dies wird von der CRA adaptiert, allerdings selektiert dieses Verfahren im Gegensatz zu anderen Ansätzen nur sogenannte „Noun-Phrases“, welche von Corman et al. definiert werden als ein Nomen gefolgt (oder geführt) von den es beschreibenden Adjektiven oder Adverben.[17] Dieser Ansatz wird deshalb gewählt, da Nomen, oder

---

[13] Vgl. Corman u.a. (2002), S. 167.

[14] Vgl. Hunter (2014), S.352

[15] Vgl. Corman u.a. (2002), S.176-177.

[16] Vgl. Hunter (2014), S. 350-366.

[17] Vgl. Corman u.a. (2002), S.174.

Substantive, den Kern einer Aussage darstellen, während Verben die Aktionskomponente verkörpern, die die Nomen miteinander verbinden und in Kontext bringen.[18] Die Nomen werden im Englischen als „nouns“ bezeichnet. Hierbei findet keine Auswahl über die Anzahl des Vorkommens eines bestimmten Begriffes statt.

2.2.2 Konzeptualisierung

Die Phase der Konzeptualisierung stellt die erste Form der Vereinheitlichung bzw. Verarbeitung der selektierten Begriffe dar. Aus dieser Phase gehen letztendlich keine Begriffe mehr vor, vielmehr dient die angesprochene Vereinheitlichung der Extraktion von Konzepten aus diesen Begriffen. Ein Konzept ist hierbei die aus dem Begriff hervorgegangene Form, welche für die weiteren Schritte der Analyse verwendet wird. Eine Form der Konzeptualisierung ist die Reduktion der Begriffe auf ihre Grundform.[17] Beispielsweise werden Begriffe wie „Engineering“ in dem Konzept „Engine“ aufgehen. Genutzt wird ein solches Vorgehen, um Begriffen einen semantischen Kontext zuzuweisen und sie hierdurch aggregieren zu können, sodass daraus gemeinsame Konzepte oder Themen ablesbar sind.[19] Die CRA beschränkt sich hier auf die Reduktion der Begriffe auf deren Singular und verzichtet auf eine Reduktion der Noun-Phrases auf deren Wortstamm. Begründet wird dies damit, dass die angesprochene Aggregation und Reduktion den Sinn der zu analysierenden Begriffe verändern kann, wodurch die folgende Ausgabe verfälscht wird.[17] Als Beispiel kann hier der Begriff „Gamification“ herangezogen werden, der auf das Konzept „Game“ reduziert würde, wodurch das eigentliche Konzept der Gamification in der finalen Ausgabe falsch zugeordnet beziehungsweise verloren gehen würde. Durch den Verzicht auf eine solche Aggregation von Begriffen ist die CRA in der Lage, die Analyse auch ohne ein Lexikon oder Referenzset von fachspezifischen Texten der Begriffe einer Domäne durchzuführen, die für das oben genannte Vorgehen vonnöten wären.[18] Deshalb eignet sich die CRA für die in dieser Arbeit durchgeführte Analyse von Abstracts wissenschaftlicher Texte, da diese häufig domänenübergreifende Begriffe und Konzepte beinhalten können.

---

[18] Vgl. Corman u.a. (2002), S.174-175.

[19] Vgl. Corman u.a. (2002), S.168-170.

2.2.3 Verbindung

Der Schritt der Verbindung wird von Hunter als „Relationship“ bezeichnet und bezieht sich auf die Basis, auf der Paare von Konzepten miteinander verbunden werden.[20] Dieser Vorgang dient der Erstellung eines Netzwerkes der identifizierten Konzepte eines Textes. Viele NTA-Verfahren verbinden Konzepte auf Basis ihres Auftretens innerhalb eines bestimmten Fensters von N Worten.[21] In diesem Fenster werden Begriffe oder die daraus mittels Konzeptualisierung erkannten Konzepte miteinander verbunden. Nachteil dessen ist, dass die größe des Fensters, oftmals über mehrere Texte nicht einheitlich ist und hierfür der semantische Kontext der Begriffe erkannt werden muss. Dafür muss dieser semantische Kontext jedoch zunächst konstruiert werden.[22] CRA nutzt hier einen von Corman et al. als „Representation“ bezeichnetes Verfahren, bei dem weder ein solches Fenster noch ein angelernter Kontext benötigt wird. Die in der vorherigen Phasen selektierten Noun-Phrases werden zunächst sequentiell nach ihrem Auftreten mit ihrem Vorgänger und Nachfolger zu einem Knoten des Netzwerks verbunden. Doppelt erfasste Begriffe werden daraufhin miteinander aggregiert und ihre Vorgänger-Nachfolger Beziehungen in einem Knoten vereint.[21] Hierdurch entsteht ein Netzwerk, welches nicht auf der Erkennung von Themen oder Konzepten einer bestimmten Domäne basiert, sondern in sich selbst die für den jeweiligen Text zentralsten Begriffe und ihre Verbindung miteinander darstellt.

2.2.4 Relevanz

Im finalen Schritt der Analyse wird die Gewichtung der einzelnen Knoten des erstellten Netzwerks berechnet, um den in den Knoten befindlichen Konzepten eine Relevanz innerhalb des ursprünglichen Textes zuzuweisen. Um einzelnen Knoten im Netzwerk eine bestimmte Gewichtung zuzuweisen existieren mehrere Metriken. Corman et al. bezeichnen diesen Schritt als „Indexing“ und nennen mehrere Möglichkeiten, dies zu berechnen.[23] Bei der „degree centrality“ wird die Anzahl der mit dem jeweiligen Knoten verbundenen weiteren Knoten gezählt, während bei der „closeness centrality“ die

---

[20] Vgl. Hunter (2014), S.350-352.

[21] Vgl. Hunter (2014), S.353.

[22] Vgl. Corman u.a. (2004), S.170-171.

[23] Vgl. Corman u.a. (2004), S.176.

durchschnittliche Anzahl der Schritte zu jedem anderen Knoten gemessen wird.[24] Im Gegensatz dazu wird im Rahmen der CRA wird die sogenannte „betweenness centrality“ angewendet, welche die Anzahl der kürzesten Wege von jedem anderen beliebigen Knoten des Netzwerkes durch jeden Knoten misst. Dabei indiziert sie auch Knoten unverbundener Netzwerke. Unverbundene Netzwerke sind Netzwerke, bei denen keine Verbindung zwischen Knoten zweier Teilnetzwerke existiert. Das Auftreten solcher Netzwerke ist bei Verwendung der CRA zur Analyse von Texten unterschiedlicher Themengebiete relativ wahrscheinlich. Diesem Knoten wird in der CRA aufgrund ihrer Zentralität besondere Bedeutung zugerechnet.[23]

3. Anforderungserhebung

Für die Analyse von Texten gibt es mehrere Vorgehen und Methodiken, die sich zum Teil stark unterscheiden. Die Auswahl einer geeigneten Methode, welche dann das Fundament der Automatisierung bilden soll, ist somit von essentieller Bedeutung. Im folgenden Abschnitt der Arbeit werden Anforderungen erhoben, auf Basis derer eine Software entstehen soll. Diese werden auch für die Auswahl eines geeigneten Verfahrens der automatischen Textanalyse verwendet.

Die Anforderungen werden zum einen auf Basis persönlicher Erfahrungen mit der Erstellung wissenschaftlicher Arbeiten erhoben und mittels Literatur belegt.

3.1 Anforderungen auf Basis wissenschaftlicher Arbeiten

Für die Erschließung des Themas einer wissenschaftlichen Arbeit ist ein fundiertes Wissen über den Themenbereich unabdinglich. Die Selektion der richtigen Begriffe für die Suche ist dabei essentiell, da die Suche durch die Wahl der Suchbegriffe bestimmt wird.[25]

Durch die Wahl der richtigen Suchbegriffe ist es dem Verfasser möglich, Literatur zu finden, die sein Domainen-Wissen erweitert und ihm einen breiteren Überblick über den Forschungsbereich verschafft.

---

[24] Vgl. Corman u.a. (2004), S.177.

[25] Vgl. Leyv, Ellis (2006), S.190., vom Brocke u.a. (2009). S. 9-10., Fink (2014). S. 26-27., sowie Wolfswinkel, Furtmueller, Wilderom (2013). S. 45-47.

Die Unterstützung des Verfassers bei der Suche ist dann gegeben, wenn die Implementierung in der Lage ist, einen Abstrakt zu analysieren und zentrale Begriffe des Abstracts zu filtern, um den Anwender einen breiteren Überblick über die Domaine seiner Arbeit zu geben. Daher lautet die erste Anforderung:

(1) Das System muss in der Lage sein, relevante Begriffe eines oder mehrerer Abstracts zu filtern.

Die nächste Anforderung basiert auf der Suche nach Literatur an sich. Für die Recherche nach wissenschaftlichen Texten werden in der Praxis Datenbanken genutzt, wo diese Texte hinterlegt sind. Dabei existiert eine Vielzahl solcher Datenbanken.[26] Nutzt man eine Datenbank für wissenschaftliche Texte, so sollten die in dieser Datenbank gespeicherten Abstracts automatisch einlesbar sein, da diese Datenbanken den Export der gefundenen Literatur ermöglichen:

(2) Das System muss Literaturdatenbanken als externe Quellen nutzen können.

Für die Entwicklung eines Prototyps wurde die Datenbank EbscoHost ausgewählt.

Die Analyse ist nur dann für einen Anwender hilfreich, wenn ihm diese auch ausgegeben wird. Die Ausgabe muss aufbereitet werden, um dem Benutzer möglichst viele Informationen zu liefern. Zentral dabei ist die Darstellung, da der Benutzer durch eine strukturierte Darstellung den größten Nutzen von der Anwendung hat. Bandara et al. beschreiben einen Ansatz, wie die Literatursuche durch Computerprogramme unterstütz werden kann.[27] Diese Unterstützung bezieht ihren Wert durch die generierten Informationen. Basierend auf der Umsetzung von CRA ist der Graph, welcher durch die NTA erstellt wird, ein wesentlicher Bestandteil der Analyse. Darüber hinaus sind die Informationen der einzelnen Noun-Phrases wie direkte Verbindungen und die berechneten Indizes für den Benutzer relevant. Diese müssen in einer für den Benutzer lesbaren Form zur Verfügung gestellt werden:

(3) Das System muss das Ergebnis der Analyse strukturiert ausgeben können.

---

[26] Vgl. Levy, Ellis (2006), S. 186-187., Vgl. Boell, Dubravka (2014). S. 278, sowie Vgl. vom Brocke u.a. (2009). S. 6.

[27] Vgl. Bandara, Miskon, Fielt (2011), S.4-5.

Levy et al. und Wolfswinkel et al. beschreiben in ihren Arbeiten Iterationszyklen, welche bei Literaturreviews durchlaufen werden.[28][29] Ein Zyklus muss abgeschlossen sein, damit der nächste beginnen kann. Dies bedeutet für das Programm, dass die Laufzeit möglichst effizient sein muss, da der Beginn eines neuen Zyklus nicht ohne die Berechung beginnen kann.

Jennex et al. empfehlen sogar, auf Reviews aufgrund des zeitlichen Aufwands zu verzichten.[30] Wenn das System mehrere Stunden für die Berechung benötigt, würde dies Jennex These stützen, dass der Review-Prozess zu aufwendig ist und damit würde das System dem Benutzer keinen Mehrwert bieten:

(4) Das System muss Import, Berechnung und Export in einer möglichst effizienten Zeit abschließen.

4. Implementierung

4.1 Generelles

Die Implementierung wurde auf Basis der in Kapitel 3 beschrieben Anforderungen erstellt. Dabei wurde das Projekt in drei Bereiche unterteilt, die zwischen den Autoren aufgeteilt wurden. Patrick Dohmen wählte ein geeignetes Framework für die Filterung von Noun-Phrases aus, integrierte dies in das Projekt und erstellte die Knoten für den Graph. Kai Augustin erstellte den Graph aus den Knoten, indem er diese entsprechen der CRA-Methode „linking“ verband. Darüber hinaus war er für die Ausführung des Programms als ganzes verantwortlich, was er durch eine umfangreiche main-Methode löste.

Adrian Kreuzer programmierte die Datenstruktur des Graphen mit Basis-Algorithmen wie Suchen innerhalb des graphen. Fortführend hat er den All-Pair-Shortest-Path-Algorhtmus entwickelt, den Kai Augustin bei der späteren Indizierung mit der Methode „Betweenes“ von CRA verwendete. Die weiteren Indizierungen wie „Centrality“ implemntierten Adrian Kreuzer und Kai Augustin zu gleichen Teilen. Kai Augustin

---

[28] Vgl. Levy, Ellis (2006), S.

[29] Vgl. Wolfswinkel, Furtmueller, Wilderom (2013), S.

[30] Vgl. Jennex (2015), S. 1.

optimierte den All-Pair-Shortest-Path Algorithmus hinsichtlich der Laufzeit und des Speicherbdedarfs.

Im Folgenden wird durch den Autor sein Beitrag für die Implementierung beschrieben. Neben einer Erklärung der Entwicklung als solche wird darauf eingegangen, welche Entscheidungen durch den Autor getroffen werden mussten und nach welchen Kriterien diese gefällt wurden.

4.2 Anteil des Autors an der Entwicklung

4.2.1 Implementierung der Datenstruktur und entsprechender Methoden

Der Autor dieser Arbeit beschäftigte sich zunächst mit einer geeigneten Datenstruktur für den Graphen, welche aus den Anforderungen (1) und (2) hervorgeht. Die gefilterten Noun-Phrases müssen für die weitere Verarbeitung gespeichert werden (1) und für das Erkennen von Verbindungen (2) müssen Methoden zum auslesen der Verbindungen zur Verfügung gestellt werden.

Der Graph musste zunächst die Möglichkeit bestitzen, die gefilterten Noun-Phrases zu speichern, was in der Variable `String nounPhrase` geschieht. Darüber hinaus musste für das „linking“ von CRA die Möglichkeit bestehen, jeden Nachbarn eines Noun-Phrases zu speichern und auch zu löschen, da es beim „mergen“ von Knoten passieren kann, dass automatisch erstellte Beziehungen wieder zu gunsten einer gemeinsamen, neuen Verbindung entfernt werden müssen (vgl. Kapitel CRA). Eine `LinkedList` vom eigenen Datentyp, in diesem Fall Elemement, bietet genau diese Möglichkeiten. Für das Entfernen eines Nachbarns wurde eine einfache Iteration über die ganze Liste erstellt, die nach einem übergebenen Element sucht und dies, wenn es gefunden wurde, entfernt.

Für das Suchen nach Knoten innerhalb des Graphen wurde ein Suchalgorithmus für Graphen implementiert, wie er in vielen Standardwerken zu Datenstrukturen vorgestellt wird. [31]

Die Klassen Graph und Element bieten mit den implementierten und beschriebenen Methoden die Basis für die Erfüllung der Anforderungen (1) und (2).

---

[31] Vgl. Cormen u.a. (2010), S. 599 – S.635.

4.2.2 Implementierung des All-Pair-Shortest-Path-Algorithmus

Hier werden die drei untersuchten Algorithmen kurz beschrieben. Eine ausführliche Beschreibung findet sich bei Cormen et al.[17] sowie den Abhandlungen von Goldberg und Radzik[32], Hougardy[33], Dijkstra[34] sowie Crauser et al.[35]

4.2.2.1 Erklärung von Floyd-Warshall

Der Algorithmus basiert auf einer Adjazenzmatrix und untersucht, welche Knoten von einem Startknoten aus erreichbar sind. Dafür itteriert der Algorithmus über die Adjazenzmatrix. Sofern Knoten C von Knoten B aus erreichar und B ein Nachfolger von Knoten A ist, wird C als Nachfolger von A gesetzt. In der Adjazenzmatrix wird als Distanz die Strecke von A nach B mit der Strecke von B nach C addiert und eingetragen.

4.2.2.2 Erklärung von Bellman-Ford

Bellman-Ford untersucht nicht die Knoten, sondern die Kanten eines Graphen. Ein Startknoten S wird gesetzt und für eine Kante X wird untersucht, ob diese bei S beginnt. Ist dies der Fall, wird die Distanz im Zielknoten Z der Kante untersucht. Ist dies größer als das Gewicht der Kante addiert mit dem in S hinterlegten Gewicht, so wird die Distanz in Z auf das Gewicht von X plus die Distanz in S gesetzt.

4.2.2.3 Erklärung von Dijkstra

Zu Beginn des Algorithmus wird die Distanz des Startknotens mit 0, die Distanz aller anderen Knoten mit unendlich initalisiert. Wenn es vom Startknoten S noch unbesuchte Knoten gibt, so wählt Dijkstra von S die Kante mit dem minimalen Gewicht aus, markiert den erreichen Knoten Z als besucht und berechnet das Gewicht des gerade gewählten Pfades. Ist dieses Gewicht kleiner, als die in Z gespeicherte Distanz, wird S als Vorgänger von Z gesetzt und die Distanz aktualisiert. Dijkstra führt diese Schritte solange aus, bis keine unbesuchten Knoten mehr vorhanden sind

---

[32] Vgl. Goldberg, Radzik (1993), S. 3-6.

[33] Vgl. Hougardy (2010), S. 279-281.

[34] Vgl. Dijkstra (1959), S. 269-271.

[35] Vgl. Crauser u.a. (1998), S. 2.

4.2.2.4 Auswahl eines Algorithmus

Basierend auf Anforderung (3) wurde durch alle Beteiligten beschlossen, unter anderem die von Corman et al. beschrieben Methode für das indizieren zu verwenden.[36]

Für den Algorithmus gab es zwei zentrale Anforderungen: Zum einen muss der Algorithmus eine möglichst effiziente Laufzeit aufweisen, um Anforderung (4) möglichst gut zu erfüllen. Nach ausführlicher Recherche stellte der Autor fest, dass APSP-Algorithmen wie Floyd und Warshall, A*, Min-Plus-Matrixmultiplikation oder Dijkstra sich zumeist in einer Laufzeit von $O(n^2)$ bis $O(n^4)$ bewegen.

Zum anderen muss der Algorithmus skalierbar sein, um Funktionen hinzufügen zu können. Die meisten ASPS gehen von einem gerichteten Graphen mit wechselnden Kantengewichten aus. Im Falle von CRA ist das Gewicht einer Kante aber immer 1 und der Graph ungerichtet.

Auf Basis dieser beiden Anforderungen wurden der Algorithmus von Floyd und Warshall, Bellman-Ford sowie Dijkstra näher untersucht.

Eine Möglichkeit war es, Floyd-Warshall zu implementieren und die Pfade mittels der Adjazenzmatrix und einer Tiefen- oder Breitensuche zu ermitteln. Dies würde die Laufzeit aber signifikant erhöhen, da neben der Pfad-Längen-Berechung auch die Pfade in einem einzelnen Schritt ermittelt werden müssten, da Floyd-Warshall nur die Länge der Pfade in der Matrix speichert. Eine Anpassung des Algorithmus auf eine Datenstruktur, die die Vorgänger beim ermitteln direkt speichert, wäre kompliziert.

Die Verwendung des Dijkstra-Algorithmus, der für einen angegebenen Startpunkt den kürzesten Weg zu allen anderen Knoten berechnet, war eine Alternative.[37] Vorteil an Dijkstra gegenüber Floyd-Warshall ist, dass Dijkstra alle Knoten auf dem kürzesten Pfad zurück gibt; diese können dann einfach in dem jeweiligen Knoten gespeichert werden. Durch die von Dijkstra erwartete und genutzte Datenstruktur ist der Algorithmus gut skallierbar, da die Datenstruktur jederzeit sich wechselnden Anforderunngen angepasst werden kann. Darüber hinaus gilt Dijkstra als der

---

[36] Vgl. Corman u.a. (2002), S. 177.

[37] Vgl. Dijkstra (1959), S.269-271.

effizienteste Algorithmus für das Singel-Source-Shortest-Path-Problem[38] (SSSP). Die kann durch eine Iteration über alle Quellen zu einem All-Pair-Shortest-Path-Problem erweitert werden. Ein Nachteil bei Dijkstra ist, dass nur ein kürzester Pfad gesucht wird. Alle weiteren kürzesten Pfade gleicher Länge werden ignoriert. Der Algorithmus muss also dahingehend angepasst werden, dass alle Pfade gespeichert werden.

Der Algorithmus von Bellman-Ford hat die selben Vorteile gegenüber Floyd-Warshall wie Dijkstra, wurde aber aufgrund der mangelnden Skalierbarkeit nicht gewählt. Darüber hinaus müsste Bellman-Ford so angepasst werden, dass die Berechnug von negativen Zyklen innerhalb eines Graphen nicht erfolgt, da dies bei Graphen von NTA nicht vorkommt. Diese Anpassung ist bei Dijkstra nicht von Nöten.

Der Autor entschied sich, Dijkstra also Algorithmus zu wählen. Um auch mehrere kürzeste Pfade speichern zu können, wurde die Datenstruktur wiefolgt erweitert:

Die Klasse Element wurde, entsprechend der Vorgaben für Dijkstra, um die Attribute minDistance, previous und ein Array adjacencies erweitert. Weiterhin wurde die Klasse Edge hinzugefügt, die die Kanten zwischen Knoten darstellt.

Eine Klasse PathSet wurde geschrieben. In dieser Klasse wird für einen kürzesten Pfad mit dem Start- und Zielpunkt gespeichert. Die Klasse Element erhielt eine LinkedList von PathSet, um alle kürzesten Pfade dieses Knoten zu jedem Anderen speichern zu können.

Da CRA jeden kürzesten Pfad braucht, wurde eine Bedingung geschrieben, die, sofern der aktuell gefundene Pfad gleich lang dem bisher kürzesten Pfad ist, den bisher kürzesten Pfad dem PathSet des Knotens hinzufügt und ihn durch den neu gefundenen Pfad ersetzt. So wird sichergestellt, dass alle kürzesten Pfade gespeichert werden. Dabei speichert die Methode den bisher kürzesten Pfad A mit Start- und Zielknoten in einem PathSet und fügt dieses dem Element des Startknotens an. Anschließend arbeitet der Algorithmus, als wenn der neu gefundene Pfad kürzer ist als der bisher gespeicherte.

---

[38] Vgl. Crauser u.a. (1998), S. 2.

Auf Basis dieser Selektion und der damit Verbundenen ersten Implementierung entwickelte Kai Augustin den Dijkstra-Algorothmus dahingehend weiter, dass er mittels Multi-Threating schneller terminierte.

5. Evaluation

5.1 Generelles

Für die Evaluation des System wurden seitens der Autoren drei Aspekte des Programms genauer untersucht.

Patrick Dohmen untersuchte das Framework für die Erkennung der Noun-Phrases, auf dessen Basis der Graph aufgebaut wurde. Kai Augustin beschäftigte sich mit der Verlässlichkeit von Abstracts als Quelle und Adrian Kreuzer untersuchte die Laufzeit des Programms. Dabei wurde nicht nur die implementierte Form getestet, sondern auch weitere Alternativen gegeneinander abgewogen.

5.2 Laufzeit des Systems mit Schwerpunkt APSP

Die von Cormen et al. vorgeschlagene Form der Indizierung basiert auf einem All-Pair-Shortest-Path Algorithmus. Diese Berechnung ist folglich extrem rechenintensiv, da der Graph merhmals durchlaufen werden muss und alle Pfade gefunden werden müssen. Laut Cormen et al. ist dies aber die beste und zuverlässigste Formel für CRA. Daher war es aus Sicht der Autoren relevant, einen Algorithmus für das ASPS-Problem zu wählen, der möglichst effizient arbeitet. Dijkstra wurde, wie bereits in Kapitel 4 beschrieben, für das System gewählt, wobei die Gründe vor allem in der Skalierbarkeit lagen. Im Folgenden wird nun die Laufzeit des Algorithmus und der Alternativen untersucht.

Vor der Auswahl von Dijkstra implementierte der Autor auf den Algorihmus von Floyd-Warshall sowie den Bellman-Ford-Algorithmus. Die Laufzeit dieser beiden Algorithmen sowie Dijkstra werden im Folgenden genauer beschrieben.

5.2.1 Laufzeit von Floyd-Warshall

Die Komplexität liegt in einem Bereich von $O(n^3)$. Bei der Implementierung erwies sich aber der Aufbau einer großen Adjazenzmatrix schon als Problem, da der Aufbau der Matrix, trotz verhältnismäßig schneller Hardware, erst nach mehreren Minuten terminierte. Da aber zum einen das System für durchschnittliche Hardware entwickelt

werden sollte und zum anderen nicht jeder Algorithmus eine Adjazenzmatrix voraussetzt, wurde der Algorithmus an dieser Stelle zurückgestellt und der Bellman-Ford-Algorithmus implementiert.

5.2.2 Laufzeit von Bellman-Ford

Die Komplexität von Bellman-Ford liegt für das CRA-Problem bei $O(n^2*m)$. Der Algorihmus wurde direkt mit Werten aus dem System getestet, wobei eine xml-Daten verwendet wurde, die mittels einer Suche bei Ebscohost erstllt wurde; diese Datei liegt dieser Arbeit aufgrund der Größe als Datei bei (cra.xml).

Bellman-Ford berechnete das All-Pair-Shortes-Path Problem in weniger als zwei Minuten, was eine signifikante Verbesserung gegenüber dem Algorithmus von Floyd-Warshall darstellte, der für den Aufbau der Matrix schon mehr als zwei Minuten benötigte. Der Autor entschied sich, die Überprüfung von negativen-Zyklen, welche Bellman-Ford ermöglicht, zu entfernen, da sie in dem betrachteten CRA-Problem nicht vorkommen können. Dies änderte allerdings nichts an der Laufzeit des Algorithmus, er terminierte weiterhin nach ungefähr zwei Minuten. Durch die Verwendung der Kanten eines Graphen muss bei der direkten Indizierung des weiteren auf eine andere Datenstruktur zugergriffen werden, in diesem Fall die Klasse Element. Das führt zu einem erhöhten Speicherbedarf und verlangsamt die Laufzeit des Algorithmus.

Die Anpassung des Algorithmus, mehrere kürzeste Pfade zu finden, erhöhte die Laufzeit dabei nicht signifikant. Die Laufzeit variierte je nach Prozessorauslastung zwischen einer und zwei Minuten. Der Spitzenwert für die Berechnung von 1968 Knoten lag bei 1 Minute und 51 Sekunden. Dies war vor allem auf die nicht optimale Datenstruktur zurückzuführen.

5.2.3 Laufzeit von Dijkstra

Die implementierte Variante von Dijkstra lies das System in maximal einer einer Minute terminieren. Die Erweiterung, dass auch mehr als ein kürzester Pfad gefunden werden kann und all jene Pfade mit gleichem Gewicht gespeichert werden, veränderte die Laufzeit nur minimal. Eine zusätzliche Iteration über die Liste der Nachfolgerknoten, welche von Nöten ist, sofern ein neuer kürzester Pfad gefunden wurde um alle bisher gespeicherten Pfade zu entfernen, lies den Algorithmus in ca.

einer Minuten terminieren. Der Spitzenwert für die Berechnung von 1968 Knoten lag bei 8 Sekunden.

Durch den direkten Zugriff auf die Knoten eines Graphen ist es bei Dijkstra möglich, die Knoten nach der Berechnung des kürzesten Weges direkt zu indizieren, was den Speicher- wie Rechenaufwand im Vergleich zu Bellman-Ford deutlich verringert.

5.3 Bewertung

Ein Austausch des Algorithmus ist aus Sicht des Autors nicht von Nöten. Für die weitere Laufzeit, welche durch das Filtern von Noun-Phrases entsteht, wird an dieser Stelle auf die Arbeit von Patrick Dohmen verwiesen.

Die Analyse von größeren Graphen anhand des Systems kann daher in nur wenigen Minuten abgeschlossen werden, abhängig von der Hardware, welche eingesetzt wird.

Die Anforderung (4) wurde daher aus Sicht des Autors vollständig erfüllt.

6. Fazit

Der Ansatz, CRA für die Automatisierung zu verwenden, erwies sich an wenigen Stellen als kompliziert. Die Grundstruktur von CRA, basierend auf NTA, und das erzeugen eines Graphen nach CRA-Vorgaben waren mit entsprechenden Frameworks gut umzusetzen. Lediglich die Implementierung der Indizierung nach Corman et al. stellt die Autoren vor Probleme, da zum einen ein APSP-Algorithmus ausgewählt und an die für das System verwendete Datenstruktur angepasst werden musste und zum anderen die von Corman et al. vorgestellte Formel umgesetzt werden musste.[39]

Betrachtet man die vier erhobenen Anforderungen, so kann festgehalten werden, dass diese fast ausnahmslos erfüllt wurden. Die Anforderungen (1), (2) und (3) wurden vollständig und ohne Einschränkung erfüllt. Für die strukturierte Ausgabe, Anforderung (2) empfehlen sich weiterführende Versuche um zu untersuchen, welche Art von Ausgabe und welches Format für einen Graphen sinnvoll wäre. Im Rahmen der Arbeit wurde die Ausgabe als .txt sowie eine umfangreiche Konsolenausgabe implementiert.

---

[39] Vgl. Corman u.a. (2002), S. 177.

Die Anforderung (4), das System in einer effizienten Laufzeit terminieren zu lassen, konnte aus Sicht der Autoren ebenfalls erfüllt werden. Dass Aufbauen des Netzwerks, welche das Filtern von Noun-Phrases sowie das Verbinden und Mergen von Knoten beinhaltet, kann durch das System effizient und schnell ausgeführt werden. Auch größere Datenmengen sind dabei in annehmbarer Zeit bewältigt worden, benötigten jedoch die größte Zeitspanne. Die Umsetzung der Indizierung der Knoten war komplex, konnte aber wie im Verlauf der Arbeit beschrieben, effizient und schnell umgesetzt werden.

Abstracts sind als Quelle für das System als verlässlich anzusehen. Im Rahmen der Evaluation von Kai Augustin konnten keine signifikanten Unterschiede in der Indizierung von Noun-Phrases von Texten und ihren Abstracts gefunden werden.

Zusammenfassend kann festgehalten werden, dass CRA durch seine Genauigkeit bei der Filterung relevanter Begriffe durchaus für die Analyse von Abstracts geeignet ist. Sieht man von dem Manko der hohen Komplexität ab, so stellt die im Verlauf dieser Arbeit erstellte Version von CRA aus Sicht der Autoren eine Bereichung bei der Literaturrecherche dar.

7. Ausblick

Die implementiere Anwendung konnte für die Zielsetzung der Arbeit zufriedenstellende Ergebnisse liefern. Ob die Anwendung auch in der Praxis sinnvoll genutzt werden kann, muss jedoch zunächst ein Anwendertest zeigen. Aus Sicht der Autoren gibt es zudem noch einige Erweiterungs und Optimierungsmöglichkeiten.

Ein möglicher Erweiterungspunkt wäre die Unterstützung weiterer wissenschaftlicher Datenbanken und ihrer Exportformate, so dass neben EBSCO-Host auch andere Datenbanken für die Abstractanalyse verwendet werden können. Weiterhin könnte auch untersucht werden, ob das Einlesen mehrerer Exportdateien möglich ist. Hierbei ist insbesondere zu untersuchen, ob eine eindeutige Identifizierung von Quellen über mehrere Datenbanken und Dateiformate möglich ist, oder ob es hier Unterschiede in den eingetragenen Daten gibt.

Als Optimierungsmöglichkeit bietet sich die Untersuchung weiterer APSP-Algorithmen an. Der Austausch des APSP-Algorithmus ist jedoch nur dann sinnvoll, wenn eine

effzientere Variante gefunden werden kann. Die Notwendigkeit hierfür ist allerdings nur dann gegeben, wenn die verwendete Formel für die Indizierung direkt eingebaut werden kann.

Des Weiteren wäre es sinnvoll, die Verwendung der Anwendung auf Texte anderer Sprachen zu überprüfen, da dies theoretisch mithilfe der Frameworks möglich und hierfür nur geringfügige Änderungen nötig wären.

Eine Erweiterung der Anwendung hinsichtlich der Benutzeroberfläche und der Export-Funktionen, vor allem Export des Graphen und der Indizierung, sind ebenfalls sinnvoll.

Literaturverzeichnis

Bandara, Wasana; Miskon, Suraya; Fielt, Erwin (2011): A systematic, tool-supported method for conducting literature reviews in information systems. 19th Eruopean Conference on Information Systems. Helsinki, 2011.

Boell, Sebastian K.; Dubravka, Cecez-Kecmanovic (2014): On being 'systematic' in literature reviews on IS. In: *Journal of Information Technology*, S. 1–13.

Corman, Steven R.; Kuhn, Timothy; McPhee, Robert D.; Dooley, Kevin J. (2002): Studying Complex Discursive Systems. Centering Resonance Analysis of Communication. In: *Human Communication Research* 28 (2), S. 157–206.

Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald; Stein, Clifford; Molitor, Paul (2010): Algorithmen - Eine Einführung. 3. Aufl. 1 Band: Oldenbourg Wissenschaftsverlag.

Crauser, A.; Melhorn, K.; Meyer, U.; Sanders, P. (1998): A Parallelization of Dijkstra's Shortest Path Algorithm. Max-Planck-Institut für Informatik. Saarbrücken. Online verfügbar unter www.researchgate.de, zuletzt geprüft am 01.07.2015.

Dijkstra, E. W. (1959): A Note on Two Problems in Connexion with Graphs, S. 269–271.

Fink (2014): Reviewing the Literature. Why? For Whom? How?

Goldberg, Andrew V.; Radzik, Tomasz (1993): A Heuristic improvement of the Bellman-Ford Algorithm. In: *Appl. Math. Lett.* 6 (3), S. 3–6.

Hougardy, Stefan (2010): The Floyd-Warshall Algorithm on Graphs with Negative Cycles. In: *Information Processing Letters*, S. 279–281.

Hunter, Starling (2014): A Novel Method of Network Text Analysis. In: *Open Journal of Modern Linguistics* 4, S. 350–366.

Jennex, Murray E. (2015): Literature Reviews and the Review Process. An Editor-in-Chief's Perspective. In: *Communications of the Association for Information Systems* 36 (2), S. 139–146.

Levy, Yair; Ellis, Timothy J. (2006): A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research. In: *Information Science Journal* 9, S. 181–212.

Michie, S.; Williams, S. (2003): Reducing work related psychological ill health and sickness absence. a systematic literature review. In: *Occup Environ Med* 60, S. 3–9.

Vom Brocke, Jan; Simons, Alexander; Niehaves, Bjoern; Reimer, Kai (2009): Reconstructing the Giant. On the importance of rigour in documenting the literature search process. European Conference on Information Systems, 2009.

Webster, Jane; Watson, Richard T. (2002): Analyzing the past to prepare for the future. Writing a Literature Review. In: *MIS Quarterly* 26 (2).

Wolfswinkel, Joost F.; Furtmueller, Elfi; Wilderom, Celeste P.M. (2013): Using grounded theory as a method for rigorously reviewing literature. In: *European Journal of Information Systems* 22, S. 45–55.

Anhang

Literatur als .zip

Test-Datei cra.xml