Bauhaus-Universität Weimar
Fakultät Medien
Studiengang Mediensysteme

Text-Reuse-Extraktion auf Basis eines Sequence-Alignment Problems

Masterarbeit

Alexander Kümmel

1. Gutachter: Prof. Dr. Benno Stein
2. Gutachter: Prof. Dr. Stefan Lucks
Betreuer: Martin Potthast

Datum der Abgabe: 29. August 2011

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Weimar, den 29. August 2011

.....
Alexander Kümmel

Zusammenfassung

Texte wiederzuverwenden ist inzwischen selbstverständlich geworden. Ob bei Nachrichten, Online-Zeitungen, Blogs und Foren, technischen Dokumentationen und Patenten. Allen sind wiederverwendete Passagen von Texten gemein. Im Rahmen dieser Arbeit wurde ein Verfahren zur Identifikation und Extraktion wiederverwendeter Texte entwickelt. Auf der Grundlage eines Sequence-Alignment Problems fließen Erkenntnisse aus der Mustererkennung in Dotplots und der dichtebasierten Clusteranalyse mit DBScan in das Verfahren ein. Die Evaluierung dieses Verfahrens zur Text-Reuse-Analyse mit einem Subkorpus des *PAN-PC* zeigt eine vielversprechende Erkennungsleistung. Vergleichende Untersuchungen zu den Teilnehmern der *PAN-Competition* und die erreichten Resultate auf dem großen *PAN-PC-10* bekräftigen diese Ansichten.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Die zunehmende Internationalisierung und Vernetzung fördert den regen Informationsaustausch und führt dazu, dass Informationen fast ständig wiederverwendet werden. Insbesondere für Nachrichtenagenturen und Zeitungen, aber auch für wissenschaftliche und wirtschaftliche Interessengruppen ist die Wiederverwendung von Informationen wichtig. In jedem der genannten Bereiche werden Fachwissen, Nachrichten zu aktuellen Ereignissen und wissenschaftliche Erkenntnisse wieder- bzw. weiterverwendet.

Journalistische Texte sind ein Beispiel dafür. Oftmals entspringen Artikel mit aktuellem Inhalt der gleichen Quelle (beispielsweise einer bestimmten Nachrichtenagentur) und werden lediglich in abgewandelter Form in verschiedenen Zeitungen veröffentlicht. Die Journalisten einzelner Zeitungen passen nachträglich die ihnen zur Verfügung stehenden Informationen den Zielgruppen und Themenschwerpunkten entsprechend an. Anschließend gelangen die bereits verwendeten Informationen in die freien Medien. An diesem Veränderungsprozess, der neue Inhalte aus bestehenden erschafft, haben Blogs und Foren einen maßgeblichen Anteil.

Vergleichbar verhält es sich mit wissenschaftlichen Veröffentlichungen und Berichten. Wissenschaft lebt von ihrer Originalität und Authentizität, aber auch von dem Drang nach Verbesserung und konsequenter Selbstkritik. Diese Qualitäten können nur durch stetige Reflexion alter und neuer Konzepte, sowie durch die Überarbeitung vorhandener Informationen gewährleistet werden. Auch in der Wissenschaft werden Informationen wiederverwendet. Durch die Kombination und Transformation vorhandener Erkenntnisse kann bestehendes Wissen zu leistungsfähigeren Konzepten weiterentwickelt werden.

Speziell das Internet fördert durch die internationale Vernetzung den Zugriff und die Verbreitung von Informationen. Mit einer stetig wachsenden Informationsmenge kann die Qualität der dargebotenen Inhalte, unter anderem aus den angesprochenen Bereichen, stark variieren. Es kann davon ausgegan-

gen werden, dass je wichtiger oder interessanter bestimmte Inhalte erscheinen, diese umso häufiger im Internet auffindbar sind. Damit ergeben sich qualitative Fragen, die die Vertrauenswürdigkeit der Quellen und der neuen Inhalte selbst betreffen. Wie kann entschieden werden, welche Information der Ursprung einer breit gefächerten Menge an Derivaten ist? Wie haben sich bestimmte Informationen mit der Zeit verändert? Wie viel Wissen tragen die neuen Informationen von ihrer ursprünglichen Quelle noch in sich? Interessant ist auch die Frage nach der Herkunft einzelner Informationsbestandteile, wie beispielsweise die Quellensuche zu einer Menge von Inhalten. Diese Fragen besitzen einen gemeinsamen Kern, denn in allen Fällen muss entschieden werden ob Informationen wiederverwendet worden sind. Folglich muss nach dem Wie und Wo von wiederverwendeten Informationen gefragt werden.

Im Rahmen dieser Arbeit werden diese Kernfragen behandelt. Dafür wird der abstrakte Begriff der Information auf Texte und Textdokumente eingeengt und in Bezug auf folgende Probleme betrachtet:

> Wie kann automatisch erkannt werden, ob Text in Dokumenten wiederverwendet wurde?
> Welche Möglichkeiten existieren, um diese gemeinsamen Textabschnitte zu extrahieren?

In dieser Arbeit wird die Wiederverwendung von Texten als *Text-Reuse* bezeichnet und jeder gefundene wiederverwendete Textabschnitt als *Text-Reuse-Instanz* verstanden.

Unter dem Gesichtspunkt der automatischen Erkennung und Extraktion wird Domänenwissen aus dem Information-Retrieval eingesetzt und mit ähnlichen Problemen fachfremder Forschungsgebiete in Verbindung gebracht. Im Kapitel 2 wird das Hintergrundwissen aufgearbeitet, das im Rahmen dieser Arbeit Verwendung findet und als Grundlage für die Entwicklung dient. Das Verfahren stellt eine Kombination aus Algorithmen des Information-Retrievals und der Bioinformatik dar. Es besitzt die Fähigkeit, in zwei gegebenen Dokumenten ähnliche Textpassagen zu erkennen und im Anschluss zu extrahieren. Vorgestellt wird die Entwicklung mit ihren Bestandteilen und spezifischen Techniken in Kapitel 3. Im Anschluss folgt in Kapitel 4 die Auswertung der evaluierten Daten und Parametereinstellungen.

Kapitel 2

Hintergrund

In diesem Kapitel stellen wir die von uns verwendeten Techniken zur Textanalyse vor. Im Hinblick auf die Algorithmusbeschreibung in Kapitel 3 betrachten wir wichtige Konzepte des Information-Retrieval. Das grundlegende Thema ist Text-Reuse. Betrachtet werden die Bedeutung und Erscheinungsformen von Text-Reuse, insbesondere in Bezug auf Identifikation und Extraktion wiederverwendeter Texte. In diesem Zusammenhang behandeln wir etablierte Konzepte des Sequenzvergleichs mit Sequence-Alignment, sowie die im Information-Retrieval häufig eingesetzte Clusteranalyse.

2.1 Information-Retrieval

Information-Retrieval (IR) ist ein Forschungsgebiet zur Entwicklung intelligenter Informationssysteme, die mithilfe modellgetriebener Konzepte Informationen verarbeiten und bereitstellen. Informationen liegen in textlicher, auditiver und visueller Form innerhalb unterschiedlicher Medien- und Dokumentformate vor. Jede dieser Formen stellt die wissenschaftliche Grundlage für abgesetzte Forschungsfelder und spezialisierte Verfahren innerhalb des IR dar. Unser Interesse konzentriert sich auf textliche Informationen und ihre Analyse.

Die Entwicklungen und Modelle des IR fließen in Information-Retrieval-Systemen (IR-Systemen) zusammen. IR-Systeme vereinen intelligente Technologien zur Unterstützung informationsbezogener Aufgaben. In diesem Sinne sind IR-Systeme überall dort anzutreffen, wo Informationen verarbeitet werden. Abhängig von der Informationsform, existieren separate Verfahren zur Strukturierung von Informationen in modellierten Repräsentationen. Nach einer Vorverarbeitung und Repräsentation der Eingangsdaten ermöglichen Informationssysteme eine inhaltsorientierte Suche, Indizierung oder Speicherung [3]. Künftig konzentrieren wir uns auf das Konzept der Informationssuche mit einem unterstützenden IR-System. Ein IR-System ist, in Bezug auf eine such-

gestützte Verwendung, ein inhaltlicher Informationsfilter, bezogen auf ein benutzerdefiniertes Informationsbedürfnis. Das Informationsbedürfnis als abstraktes Konzept muss zur Verarbeitung durch den Computer in einem konkreten Anfragemodell definiert werden. Der Benutzer formuliert eine oder mehrere Anfragen (*Queries*) und beschreibt damit sein Informationsbedürfnis in einer für das IR-System verständlichen Form. Abhängig von der Komplexität des Informationsbedürfnisses kann eine Query aus Wörtern, Phrasen oder größeren Textabschnitten und Texten bestehen. Das IR-System benutzt die formulierten Queries zur Suche relevanter Daten innerhalb seiner Datenbasis.

2.1.1 Relevanz und Retrieval

Ein suchbasiertes IR-System wird darauf hin konzipiert, die Dokumente einer Datenbasis hinsichtlich ihrer Relevanz zu einer Query zu bewerten. Die Zuordnung relevanter Informationen soll dabei mit minimalen Fehlern gewährleistet werden [4]. Informationen können als relevant angesehen werden, wenn die Inhalte Aussagen über das nachgefragte Wissen enthalten [13]. In diesem Sinne sind die Informationen zweckmäßig und nützlich für den Benutzer. Die Relevanzentscheidung basiert im Information-Retrieval unter anderem auf der strukturellen Übereinstimmung von Query und Dokument. Wir nehmen an, dass strukturelle Ähnlichkeit auch inhaltliche Ähnlichkeit widerspiegelt. Wir betrachten Relevanz als ein Maß der Ähnlichkeit zwischen einem Dokument und einer Query. Zur automatisierten Verarbeitung von Query und Dokument ist es notwendig, die vorhandenen und nachgefragten Informationen in einem Modell abzubilden.

2.1.2 Das Retrieval-Modell

Das Retrieval-Modell repräsentiert die Daten und definiert die damit verbundene Verarbeitung. Dadurch können, abhängig von einer konkreten Aufgabenstellung, Retrieval-Modelle passend zu den Anforderungen gewählt werden. Die Eingangsdaten werden dafür zunächst innerhalb des Retrieval-Modells durch einheitliche Datenstrukturen abstrahiert. Diese Dokumentrepräsentationen orientieren sich abhängig vom Modell an unterschiedlichen Merkmalen der Eingangsdaten. Welche Merkmale hier zum Einsatz kommen können, werden wir später betrachten.

Zur Verarbeitung einer Dokumentrepräsentation $\mathbf{d}$ aus der Datenbasis mit einer Query $\mathbf{q}$ definiert das Retrieval-Modell $\mathcal{R}$ die Funktion $\rho_{\mathcal{R}}(\mathbf{q}, \mathbf{d})$. Diese Retrieval-Funktion macht die Relevanz eines Dokuments zur Query messbar und kann sich aus verschiedenen Kriterien zusammensetzen. Zur Bewertung der Relevanz kommen unter anderem Ähnlichkeitsmaße und Distanzmaße zur

messbaren Textähnlichkeit zum Einsatz. Die Ergebnisse der Retrieval-Funktion werden als Retrieval-Werte bezeichnet [38].

$$\rho_{\mathcal{R}} : Q \times \mathcal{D} \rightarrow \mathbb{R} \tag{2.1}$$

Diese müssen während des Retrievals für alle Kombinationspaare aus Queries und den Dokumenten der Datenbasis berechnet werden (2.1). Wir betrachten die Retrieval-Werte als ein zentrales Bewertungskriterium für die Relevanz eines Dokuments zum Informationsbedürfnis des Benutzers.

2.1.3 Retrieval-Task

Es ist offensichtlich, dass mit unterschiedlichen Anforderungen an IR-Systeme auch die notwendige Informationsverarbeitung variiert. Der Retrieval-Task definiert die Anforderungen als informationsbezogene Aufgabe an das IR-System. Abhängig von der Komplexität der Aufgabe sind IR-Systeme unterschiedlich gut geeignet, ein Informationsbedürfnis zu erfüllen.

Ein Retrieval-Task ist beispielsweise die Suche nach Dokumenten im Internet. Das IR-System ist in diesem Fall eine Suchmaschine. Die Query kann aus einzelnen Suchtermen bestehen. Die Datenbasis wird durch die von der Suchmaschine indizierten Dokumente gebildet. Jedes Dokument der Datenbasis ist im einfachen Fall als Menge der enthaltenen Wörter modelliert. Die Gesamtheit dieser Relationen ist der Index der Suchmaschine. Die Relevanz wird im naiven Fall durch den direkten Vergleich der einzelnen Suchterme mit den Stichwörtern der Query bestimmt. Sie steigt abhängig von der Anzahl der übereinstimmenden Terme.

Mithilfe eines vorher definierten Retrieval-Tasks kann das konkrete IR-System, wie im Beispiel, gewählt werden. Dessen Datenmodell und Relevanzkriterium dient im Anschluss der Festlegung eines Verarbeitungsprozesses. Demzufolge ist der Retrieval-Task essenzieller Teil für die Definition einer Analyse von Textdokumenten. Alle notwendigen Mittel können durch diese Anforderungsbeschreibung bestimmt werden. Die Informationen zur konkreten Analyse, in unserem Fall der Textanalyse auf Textdokumenten, bestimmen sich durch die Extraktion und Verarbeitung von Texteigenschaften.

2.1.4 Texteigenschaften

Informationsaggregation aus unterschiedlich strukturierten und unstrukturierten Textdaten bedingt die Auswahl bestimmter messbarer Textmerkmale zur Informationsverarbeitung. Abgesehen vom Aufwand einer semantischen Analyse können strukturelle Textmerkmale eines Dokumentes hinreichende Informationen für das Retrieval beitragen. Verschiedene Merkmale können, abhängig

von einem Retrieval-Modell, in einer definierten Merkmalsmenge zusammengefasst und kombiniert werden. Strukturelle Merkmale sind bei Textdokumenten unterschiedlich ausgeprägt. Ohne eine konkrete Merkmalsausprägung kann diese Information später nicht genutzt werden. Aus diesem Grund werden die Merkmale nicht nur unterschiedlich in einer Textanalyse gewichtet, sondern auch entsprechend ausgewählt. Generische strukturelle Merkmale sind beispielsweise die Textlänge und die mittlere Satz- und Wortlänge. Dazu zählt ebenso die Wortfrequenz, allgemein oder bezogen auf bestimmte Worttypen, in Relation zum aktuellen Dokument oder bezogen auf die Menge aller Dokumente einer vorgegebenen Menge. Diese und weitere Eigenschaften werden an anderer Stelle [5] detaillierter betrachtet.

2.1.5 Textähnlichkeit

Die Quantifizierung der Texteigenschaften wird in der Textanalyse genutzt, um die Ähnlichkeit zwischen Texten zu bewerten. Nach unserem Verständnis beruht Textähnlichkeit auf einer syntaktischen Analyse der Eingangsdokumente. Der paarweise Vergleich zweier Dokumente d_1 und d_2, wird durch die Dokumentrepräsentationen $\mathbf{d}_1$ und $\mathbf{d}_2$ ermöglicht. Dafür ist die Definition einer Distanzfunktion $\delta(\mathbf{d}_1, \mathbf{d}_2) \rightarrow [0, 1]$ notwendig. Diese Funktion vergleicht zwei Dokumentrepräsentationen eines Retrieval-Modells $\mathcal{R}$ aufgrund vorgegebener Merkmale und Parameter miteinander. Die Distanzfunktion gibt eine Aussage darüber, wie unähnlich sich zwei Texte sind. Die Ähnlichkeit $\varphi(\mathbf{d}_1, \mathbf{d}_2)$ zwischen diesen Texten wird über die Beziehung $\varphi(\mathbf{d}_1, \mathbf{d}_2) = 1 - \delta(\mathbf{d}_1, \mathbf{d}_2)$ hergestellt [6].

Es existieren bereits verschiedene Distanzfunktionen, die auf unterschiedlichen Eigenschaften aufbauen und abhängig vom Retrieval-Task eingesetzt werden. Die folgenden Beispiele verdeutlichen unterschiedliche Möglichkeiten zur Definition von Textähnlichkeit.

Die Levenshtein-Distanz, oder auch Edit-Distance, misst die Unähnlichkeit zwischen zwei Textfragmenten über die Anzahl der notwendigen Änderungsoperationen eines Fragments bis hin zur Identität mit dem anderen Fragment. Die Ähnlichkeit zwischen den Textfragmenten ist umgekehrt proportional zur Anzahl der notwendigen Änderungen. Erlaubte Operationen bei diesem Verfahren sind das Ersetzen, Löschen und Einfügen von Zeichen [6].

Werden indes die unterschiedlichen, nicht notwendigerweise aufeinanderfolgenden Zeichen zwischen zwei Textfragmenten entfernt und ist dies die einzige zulässige Veränderung, so ergibt sich nach dieser Operation die größte gleichgerichtete Menge gemeinsamer Buchstaben beider Textfragmente, die sogenannte Longest-Common-Subsequence [6].

Vektorbasierte Retrieval-Modelle nutzen Ähnlichkeitsmaße innerhalb mehrdimensionaler Vektorräume. Jedes Dokument wird innerhalb dieser Retrieval-

Modelle als Vektor repräsentiert. Die Anzahl der Merkmale bestimmt die Dimensionalität des Vektors, dessen Komponenten die einzelnen Merkmalsausprägungen enthalten. Über den Kosinus des eingeschlossenen Winkels zwischen zwei Vektoren wird die Ähnlichkeit der Dokumentrepräsentationen zueinander bestimmt. Die abstrahierte Textähnlichkeit wird deshalb als Kosinusähnlichkeit bezeichnet.

Zusätzlich zu den bisher genannten Varianten zur Bestimmung der Ähnlichkeit und ihrer Abbildung auf einen festen Wertebereich existieren noch weitere Möglichkeiten. Hash-basierte Verfahren bilden Textähnlichkeit auf einen binären Wertebereich ab [36]. Grundlage ist die Erzeugung von Hashwerten für Textdokumente oder Textabschnitte anhand verfügbarer Textmerkmale. Die Zusammensetzung der Hashwerte kann abhängig vom Verfahren variieren. Die Ähnlichkeitsbestimmungen erfolgen durch Hashwertkollisionen. Kryptografische Hashwerte bilden eine immer gleiche Eingabe eindeutig auf einen Hashwert ab. Kollisionen bei einem Hashwertvergleich identifizieren damit strukturell identische Textabschnitte. Weitere Entwicklungen im Bereich der Fuzzy-Systeme nutzen Hashwerte aus mehreren Merkmalskombinationen. Diese Fuzzy-Hashwerte [35] werden mit der Zielsetzung berechnet, die strukturelle Verschiedenheit semantisch ähnlicher Text über gleiche Hashwerte zu kompensieren. Die differenzierte Bewertung der Textähnlichkeit wird durch den Hashwertvergleich, mit einer möglichen Kollision, auf eine eindeutige binäre Antwort nach Ähnlichkeit oder Unähnlichkeit abgebildet.

Textähnlichkeit ist entscheidend in der Bestimmung textlich relevanter Informationen während des Information-Retrievals. Unterschiedliche Informationssysteme nutzen diesen Grundbaustein für ihre Textanalyse und setzen dafür auf eine ähnliche Dokumentvorverarbeitung. Wir setzen zur Bestimmung der Textähnlichkeit ein komplexes System ein, das in Kapitel 3 näher beschrieben wird. In Bezug auf das von uns entwickelte Verfahren betrachten wir zusammenfassend die essenziellen Vorverarbeitungsschritte für Textdokumente.

2.1.6 Dokumentvorverarbeitung

Die Textanalyse unterschiedlicher Dokumente setzt eine einheitliche Vorverarbeitung der Dokumente voraus. Um für den Retrieval-Prozess verwertbare Informationen aus den Eingabedokumenten zu extrahieren, müssen die Textdokumente zunächst normalisiert werden. Die Normalisierung sorgt für wohldefinierte Ausgangsvoraussetzungen und ist integraler Bestandteil jeder Dokumentvorverarbeitung [7]. Danach folgt die Repräsentation durch das Retrieval-Modell und die Verarbeitung anhand des spezifischen Retrieval-Tasks. Die fol-

gende Auflistung setzt sich aus den für uns wichtigen Schritten zur Textnormalisierung zusammen.

Lexikalische Analyse

Die Struktur der Eingabedokumente ist einem IR-System zunächst unbekannt, da die Texte als Fließtext vom System gelesen werden. Aufgrund dessen muss die logische Struktur der Textbestandteile zunächst identifiziert und abhängig vom Retrieval-Task wiederhergestellt werden. Damit verbunden ist das als Tokenisierung bekannte Verfahren zur Rekonstruktion lexikalisch relevanter Informationen, wie Wörter, Sätze und komplexere Textstrukturen. Die Behandlung von Satz- und Sonderzeichen innerhalb von Wörtern, sowie die Identifizierung sogenannter Entitäten wie Daten, Jahreszahlen und Währungen, fällt in den Aufgabenbereich verschiedener Tokenisierungsverfahren.

Entfernen der Stoppwörter

Stoppwörter sind vom Menschen definierte Wörter, die während des Retrieval-Prozesses aus dem Datenstrom der Textdokumente gefiltert werden. Dafür existieren Stoppwortlisten, die sich meist aus Funktionswörtern und häufig auftauchenden Wörtern einer Sprache zusammensetzen, jedoch nicht darauf festgelegt sind. Die Stoppwortentfernung wird unter der Annahme verwendet, dass diese Worte nur wenig Information zum Inhalt beitragen. Zusätzlich kann durch die Entfernung von Stoppwörtern die Datenmenge aller Dokumente der Datenbasis erheblich reduziert werden.

Ersetzungen aus Thesauri

Diese Normalisierungsstufe ersetzt Wörter durch ihre Synonyme. Abhängig vom vorgegebenen Verhalten variiert die Art und Weise, welche Wörter durch ihr jeweiliges Bedeutungsäquivalent ersetzt werden. Die beiden offensichtlichsten sind die Ersetzung durch das am häufigsten genutzte Synonym oder das seltenste, abhängig von der Anzahl der Alternativen. In jedem Fall muss die Normalisierung einheitlich für jedes vorkommende Wort durchgeführt werden. Aufgrund der Reduktion auf eine einheitliche Variante ergibt sich eine gleichlautende Sachbezeichnung. Jedoch muss beachtet werden, dass die originalen Wortformen und damit weitere Informationen verloren gehen können.

Stemming

Als Stemming werden die Methoden zur Rückführung eines Wortes auf seine Stammform bezeichnet. Die Technik kommt zur Verbesserung der Effektivität

und der Datenreduktion während des IR-Prozesses zum Einsatz [15]. Es existieren unterschiedliche Ansätze, die sprachabhängig oder allgemein arbeiten. Regelbasiertes Stemming oder statistisches Stemming auf Basis eines Sprachmodells gehören zu den komplexen Verfahren dieser Kategorie. Weiterhin existieren einfache Heuristiken, die lediglich eine feste Zeichenzahl als Suffix oder Präfix von jedem Wort abtrennen bzw. beibehalten.

Chunking

Texte besitzen oft unterschiedlich lange Sätze, Absätze und Passagen. Dies erschwert die einheitliche Verarbeitung während einer Textanalyse. Häufig wird unterschiedlichen Textlängen entgegen gewirkt, in dem die Texte in Fragmente fester Länge zerlegt werden. Die Länge wird abhängig von der gewählten Repräsentation der Texte in der Anzahl der Wörter oder in der Anzahl der Zeichen ausgedrückt. Die beiden Varianten werden als Wort-N-Gramme und Zeichen-N-Gramme bezeichnet, deren Länge durch N angegeben wird. Der Vorgang, diese Wort- und Zeichengruppen aus einem Fließtext zu extrahieren, wird als Chunking bezeichnet. Neben diesen grundlegenden Varianten können noch weitere Kombinationen auftreten, auf die an dieser Stelle nicht weiter eingehen werden soll.

Diese grundlegenden Verfahren zur Textnormalisierung können in ähnlicher Form in jedem IR-System eingesetzt werden. Sie helfen dabei, die menschenlesbaren Textdokumente für eine einheitliche Dokumentrepräsentation vorzubereiten. In dieser Hinsicht untermauern sie das Datenmodell durch einheitlich verwertbare Textinformationen als unerlässliche Grundlage für eine Textanalyse.

2.2 Text-Reuse

Text-Reuse umfasst die Wiederverwendung von Texten in neuen Texten mit dem Ziel, die vorhandene Information zu erhalten und weiterzugeben. Text-Reuse ist ein Oberbegriff für eine Reihe von Texttransformationen, die unter dieser Prämisse funktionieren. Generell ordnen sich hier die unterschiedlichen Kommentier- und Berichtsformen ein, sowie die bekannten Zitiermöglichkeiten und -vorgaben. Ausgeprägtere Formen von Text-Reuse erwachsen aus der Modifikation von Texten bezogen auf ihre Ansprache, das Medium und die Klientel der Betrachter. Dabei wird der Inhalt weitgehend auf eine bestimmte Verwendung angepasst, wobei die Kernaussagen erhalten bleiben. Diese Form des Text-Reuse ist als Paraphrasieren bekannt. Beispiele für Text-Reuse finden

sich unter anderem in den Bereichen Marketing und Journalismus. Hier werden die Texte zur zielgruppengerechten Präsentation umformuliert. Vor allem die Texte in Zeitungen und Online-Medien sind geprägt von Text-Reuse.

Unter die praktische Verwendung von Text-Reuse fallen auch Patente, Gesetzestexte und unterschiedliche Formen von Anleitungen. Insbesondere technische Dokumentationen und Anleitungen werden häufig mit unterstützenden Technologien [31] aus dem Forschungs- und Anwendungsbereich des technischen Schreibens erstellt. Die Wiederverwendung von Texten findet hier durch wohldefinierte Regeln zur erneuten Platzierung in Materialien und für unterschiedliche Geräte und Medienformate ihre Anwendung. Spezialisierte Systeme fördern diese Art des Schreibens durch Verwaltung und Organisation der Textinhalte und Regelsätze.

2.2.1 Charakterisierung

Die Text-Reuse-Analyse hebt sich von anderen Analyseverfahren zur Bestimmung ähnlicher Texte ab. Die etablierten Verfahren zur Near-Duplicate-Detection und Plagiatanalyse arbeiten überwiegend auf Dokumentebene [24]. Die Text-Reuse-Analyse hingegen baut auf wesentlich kleineren Informationseinheiten auf. Dies können Wortgruppen, Sätze und Absätze umfassen, die in umformulierten, zitierten und kommentierten Varianten erneut Verwendung finden. Diesbezüglich wurde Text-Reuse bereits in früheren Arbeiten [12] in einem Ähnlichkeitsspektrum eingeordnet. Das Spektrum (Abb. 2.1) erstreckt sich zu beiden Grenzen hin über den im Information-Retrieval nutzbaren Möglichkeitenraum zur Identifikation relevanter Dokumente. Die beiden Extrema sind die absolute Identität und die thematische Ähnlichkeit zweier Textdokumente.

Identität ist das stärkste Kriterium für Ähnlichkeit, nur strukturell und semantisch gleiche Dokumente sind relevant. In diesem Fall ist ein Dokument die Query an das IR-System. Variationen dieses Prinzips erlauben auch Abwandlungen der Query. An dieser Stelle dürfen strukturelle Abweichungen jedoch nicht den semantischen Themenbezug verändern. Die Verfahren zur Near-Duplicate-Detection sollen diese Variabilität identifizieren und den Grad dessen als Ähnlichkeitsmaß quantifizieren.

Thematische Ähnlichkeit auf der anderen Seite des Spektrums stellt sich während der Suche nach Dokumenten ein, wenn lediglich themenspezifische Schlagworte als Query an das IR-System übergeben werden. Suchmaschinen arbeiten nach diesem Konzept und vernachlässigen dabei meist die Struktur der gefundenen Dokumente. Unabhängig von themenspezifischen Texten oder nachrichtenartigen Teasern mit gemischten Themen, wird das Dokument als relevant eingestuft, sobald die Bestandteile der

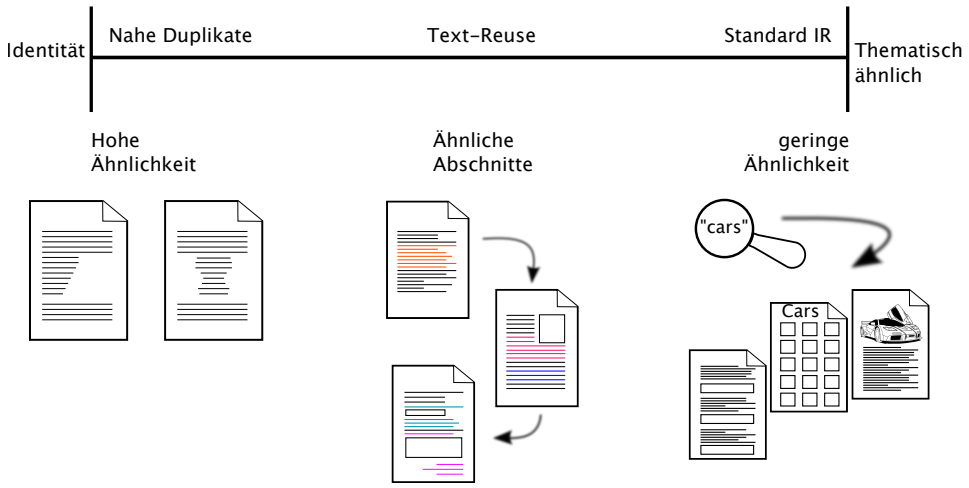


Abbildung 2.1. Eine Einordnung der bekannten Standardsuchverfahren im IR (rechts), der Suche nach nahen Duplikaten und sehr ähnlichen Dokumenten (links) und der Mischform Text-Reuse mit ihrer nur abschnittsweisen Ähnlichkeit der Textbestandteile (mittig).

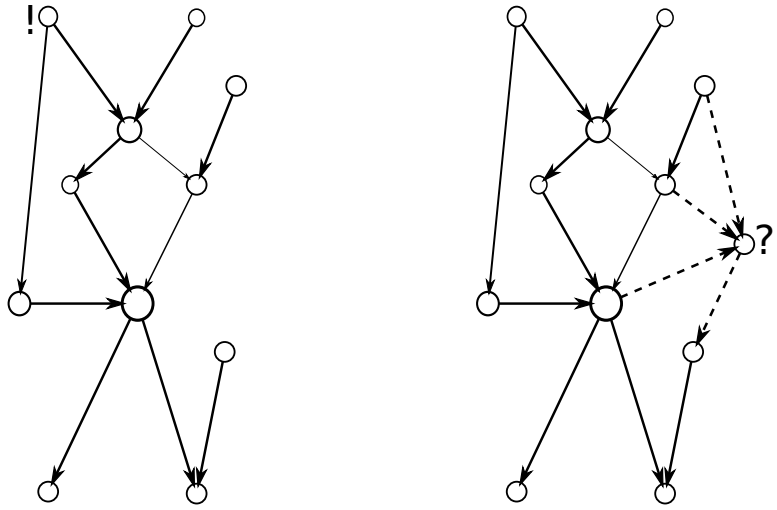
Query mit denen des Dokuments übereinstimmen.

Text-Reuse wird im Mittelfeld des Spektrums eingeordnet. Diese Tatsache ist vor allem darin begründet, dass Text-Reuse als Wiederverwendung stark fragmentierter Textbestandteile charakterisiert werden kann. Genauer gesagt bedeutet dies, dass in der Textanalyse die wiederverwendeten Teile aus anderen Texten deutlich kleiner ausfallen. Textabschnitte können durch Texttransformationen in ihrer Ansprache, ihrem Layout und ihren Inhalten verändert werden. Charakteristisch für Text-Reuse ist die Beständigkeit der Information über die Transformationen hinweg.

2.2.2 Verwandte Forschung

Eine Weiterführung der Text-Reuse-Analyse ist die Information-Flow-Analyse [25]. Hier werden Fragen nach der zeitlichen Entwicklung von Konzepten und Informationen aufgeworfen. Darunter fällt auch die grundsätzliche Erkennung von Text-Reuse über mehrere Transformationen hinweg und die entsprechende Extraktion der Textbestandteile. Die Information-Flow-Analyse ist zum Beispiel dann von Bedeutung, wenn öffentliche Ereignisse Einfluss auf die allgemeine Berichterstattung ausüben und sich Prioritäten in den Medieninformationen verschieben.

Dazu lässt sich ein Vergleich zwischen der Text-Reuse-Analyse und der häufig, fälschlicherweise gleichgesetzten Plagiatanalyse ziehen. In Abbildung 2.2(a) wird das Text-Reuse-Problem als Graphenproblem veranschaulicht. Die



(a) Text-Reuse-Analyse (b) Plagiatanalyse

Abbildung 2.2. Text-Reuse-Analyse und Plagiatanalyse abstrahiert als Graphenproblem. Text-Reuse-Analyse mit einem vorgegebenen Text (!) und der Aufgabe, Derivate zu finden. Plagiatanalyse mit der Aufgabe, die Quellen eines verdächtigen Dokuments (?) zu identifizieren.

Knoten des Graphen sind eine Menge von Texten, die wiederverwendete Informationen enthalten können. Ausgehend von einem Anfang, dem ersten dokumentierten Auftauchen einer Information, bis hin zu all ihren Variationen, werden diese im Graphen abgetragen und sind über Kanten miteinander verbunden. Die Text-Reuse-Analyse auf einer Menge von Texten besteht nun darin, diesen Graphen aufzubauen.

Die Plagiatanalyse arbeitet auf einer Menge von Texten, die der Annahme nach Informationen teilen. Hier wird der Graph als Menge dieser Textknoten vorgegeben (Abb. 2.2(b)) und die Aufgabe besteht darin, einen weiteren Text in diesen Graphen einzuordnen. Lässt sich die Zuordnung erfolgreich durchführen, dann ist der Text als Derivat der Texte im Baum identifiziert. Die Plagiatanalyse liefert hauptsächlich die direkten Nachbarn des gesuchten Textes als Ergebnis zurück.

2.3 Clusteranalyse

Die Clusteranalyse ist ein Werkzeug zur automatischen Kategorisierung von Objekten anhand bestimmter Merkmale in homogene Gruppen. Die Objekte einer Gruppe sollen dabei Homogenität untereinander aufweisen, aber hetero-

gen gegenüber den Objekten anderer Gruppen sein [16, 21].

Bei der Verarbeitung von Textdokumenten treten Datenmengen auf, deren unterschiedliche Merkmalsausprägungen keine klare Sicht auf die Zusammenhänge zulassen. Die Clusteranalyse ist in diesem Fall vor allem dann angebracht, wenn die Zusammenhänge zwischen den Eingabedaten und den Merkmalsausprägungen noch unbekannt sind. Wir sprechen bei den Eingabedaten von einer Dokumentmenge mit einzelnen Textdokumenten. Für eine beliebige Dokumentmenge kann mit Clusteranalyseverfahren eine Struktur hergestellt werden. Ziel der Clusteranalyse ist die unüberwachte und automatische Strukturierung der Dokumente einer Dokumentmenge durch Kategorisierung in merkmalsabhängige Teilmengen. Wichtige Entscheidungen sind die Auswahl der Textmerkmale, der zu verwendende Clusteralgorithmus und das Ähnlichkeitsmaß [17]. Voraussetzung für die Klassifizierung ist eine einheitliche Repräsentation der Dokumente, die deren Vergleichbarkeit gewährleistet.

Der Prozess der Clusteranalyse wird als Clustering bezeichnet. Sei D eine Dokumentmenge mit $D = \{d_1, \ldots, d_n\}$. So ist ein (exklusives) Clustering $\mathcal{C}$ von D, $\mathcal{C} = \{C_1, \ldots, C_m\}$, $C_i \subseteq D$, eine Aufteilung von D in paarweise disjunkte Mengen C_i mit $\bigcup_{C_i \in \mathcal{C}} C_i = D$ [37]. Die Dokumente eines Clusters besitzen in einem Clustering einen hohen Zugehörigkeitsgrad gegenüber Dokumenten innerhalb des gleichen Clusters und eine niedrige Zugehörigkeit gegenüber den Dokumenten anderer Cluster [16]. Jedes Cluster C_i beschreibt eine Klassenzugehörigkeit der enthaltenen Dokumente aufgrund ihrer Merkmale.

Damit erlauben die Resultate der Clusteranalyse, Rückschlüsse auf Merkmalsausprägungen und -tendenzen zu ziehen, sowie die Beziehungen zueinander auszuwerten. Im Information-Retrieval wird die Clusteranalyse zur Verbesserung der Retrieval-Ergebnisse eingesetzt. So können Dokumente beispielsweise anhand der enthaltenen Wörter oder basierend auf einer Query kategorisiert werden.

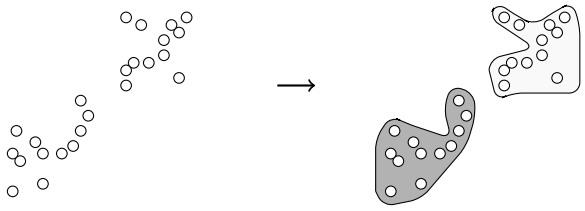


Abbildung 2.3. Clusteranalyse am Schema. Die Eingabedaten links und die durch das Clustering erkannten Gruppierungen rechts.

Am Beispiel in Abbildung 2.3 wird eine Datenmenge, bestehend aus den für einen Menschen ersichtlichen Häufungen, bezüglich eines Merkmals in ein Clustering von zwei Clustern aufgeteilt. Die Clusteranalyse ist in diesem Sinne

```
L  G  P  S  S  K  Q  T  G  K  G  S  --  S  R  I  W  D  N
|           |     |  |  |  |           |        |
L  N  --  I  T  K  S  T  G  K  G  A  I  M  R  L  G  D  A
```

(a) Globales Alignment

```
L  G  P  S  S  K  Q  T  G  K  G  S  S  R  I  W  D  N
                   |  |  |  |
   L  N  I  T  K  S  T  G  K  G  A  I  M  R  L  G  D  A
```

(b) Lokales Alignment

Abbildung 2.4. Vergleich zwischen globalem und lokalem Alignment, anhand einer Zeichensequenz aus der Gensequenzanalyse [26].

eine Annäherung an die menschliche Erkennungsleistung. Konkrete Clusterverfahren werden dafür entwickelt und getestet, um dem beschriebenen Clustering weitestgehend nahezukommen.

2.4 Sequence-Alignment

Als Basistechnologie vieler Textanalysesysteme sind Verfahren zum Vergleich und zur Extraktion der Textbestandteile entwickelt wurden. Darauf aufbauend konnten Metriken, statistische Merkmale und andere Messwerte abgeleitet werden. Das Sequence-Alignment ist eine dieser Technologien zur Textanalyse. Es hat seinen Ursprung in der Bioinformatik [10] und weist Parallelen zum String-Alignment aus anderen Gebieten der Textanalyse auf. Das generelle Vorgehen kann als Mustererkennung in Zeichenketten betrachtet werden. Sequence-Alignment wird in der Bioinformatik zur Gensequenzanalyse definiert als ein Prozess, bei dem mindestens zwei Zeichensequenzen paarweise oder vielfach miteinander verglichen werden [26]. Der Vergleich dient zur Identifizierung gemeinsamer Zeichen, die in gleicher Reihenfolge in beiden Sequenzen auftreten.

Zu unterscheiden sind die beiden grundlegenden Varianten globales und lokales Sequence-Alignment. Beide definieren paarweise Sequenzvergleiche, die auf einer Betrachtung der Einzelzeichen beruhen. Der Vergleich mehrerer Zeichensequenzen gleichzeitig [27] leitet sich aus den vorgestellten Varianten ab, weist jedoch erheblich komplexere Strukturen auf und dient in diesem Fall nicht dem grundlegenden Verständnis des Verfahrens. Aus diesem Grund wird an dieser Stelle nicht weiter darauf eingegangen.

Globales Alignment

Sind zwei Zeichensequenzen s_1 und s_2 gegeben, wird durch globales Alignment die Anzahl übereinstimmender Zeichen beider Sequenzen maximiert. Erlaubt sind bei diesem Zeichenvergleich die Zustände Übereinstimmung, Unterschied und Aussparung. Abhängig von der Länge beider Zeichensequenzen wird für beide die Möglichkeit eingeräumt, Aussparungen oder auch Löcher in begrenztem Maße zuzulassen, um mit der gegenüberliegenden Sequenz an anderer Position besser übereinzustimmen. Die Abbildung 2.4(a) veranschaulicht dieses Vorgehen, das zu den Enden beider Sequenzen hin fortgesetzt wird, um eine maximale Übereinstimmung zu gewährleisten.

Lokales Alignment

Lokales Alignment vergleicht zwei Zeichensequenzen s_1 und s_2, indem es die beste Übereinstimmung zweier Teilsequenzen $\alpha \in s_1$ und $\beta \in s_2$ findet. Dies geschieht durch Bewertung aller möglichen paarweisen Kombinationen von α und β, sowie durch die Maximierung übereinstimmender und aufeinanderfolgender Zeichen in α und β. Bei diesem Prozess nehmen α und β jeweils unterschiedliche Ausprägungen an, wodurch sich die Variationsvielfalt erneut erhöht. Lokales Alignment priorisiert Bereiche mit hoher Dichte identischer Zeichengruppen. Dadurch entstehen in den Resultaten des Sequenzvergleichs oft inselartige Gruppierungen unterschiedlicher Größe. Im Beispiel in Abbildung 2.4(b) ist deutlich zu sehen, dass trotz der Übereinstimmung einzelner Zeichen an anderen Sequenzpositionen, nur der mittlere Bereich als finales Ergebnis anerkannt wird. Lokales Alignment ist, durch diese Priorisierung auf lokale Detailbereiche, unabhängig von der Länge der Vergleichssequenzen.

Kapitel 3

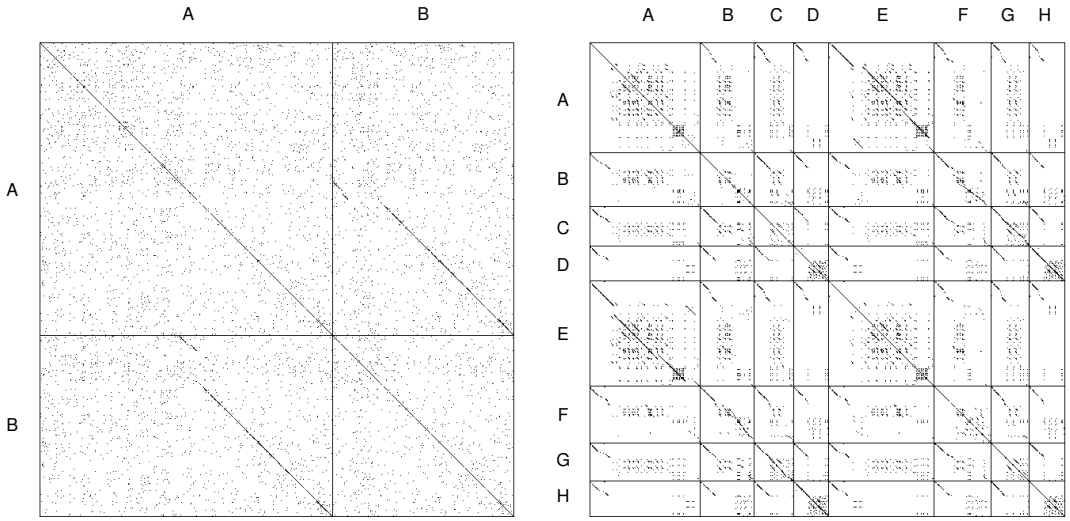
Text-Reuse-Extraktion: Clusteranalyse auf Dotplots

In diesem Kapitel stellen wir das von uns entwickelte Verfahren zur Identifizierung und Extraktion wiederverwendeter Textbestandteile vor. Dafür werden zunächst die technologischen Komponenten betrachtet, die wir als Werkzeuge zur Textanalyse einsetzen. Mit diesem Vorwissen betrachten wir im Anschluss die konkrete Anwendung unserer Dokumentverarbeitung und die darauf folgende Datenanalyse.

3.1 Mustererkennung und Clusteranalyse

Die eingesetzten technischen Verfahren leiten sich aus generischen Werkzeugen zur Erkennung von Mustern und Identifikation von Klassen ab. Der Einsatz dieser Technologien ist nicht beschränkt auf das Information-Retrieval und hat teilweise seinen Ursprung in artverwandten Forschungsbereichen, wie der Mustererkennung und Ähnlichkeitsklassifizierung von Genen und Gensequenzen in der Bioinformatik [22]. Unsere Entwicklung unterscheidet sich von anderen Verfahren im Umfeld der Text-Reuse-Analyse hinsichtlich des Aufbaus der Textanalyse.

Wir wollen zunächst auf die von uns verwendete grundlegende Idee der Mustererkennung in Zeichensequenzen eingehen. Als wichtiges Konzept in unserem Verfahren, ist ihre Verwendung in der Textanalyse näher zu betrachten. Im Anschluss stellen wir das von uns genutzte Clusterverfahren DBScan vor und erläutern seine Funktionsweise.



(a) Eine Gensequenz im Selbstvergleich.

(b) Ein Vergleich von 3000 Zeilen Quelltext.

Abbildung 3.1. Dotplots als visueller Vergleich zweier unterschiedlicher Zeichensequenzen [11]. Die Bereiche sich wiederholender Sequenzen sind durch Buchstaben und Quadranten gekennzeichnet. Ein Punkt entspricht einem identischen Fragment der zu beiden Achsen abgetragenen Zeichensequenzen. Abhängig von der Definition können Fragmente beispielsweise aus Wort-N-Grammen, Zeichen-N-Grammen und Wörtern bestehen.

3.1.1 Manuelle Mustererkennung mit Dotplots

Als Dotplots wird eine Gruppe von Verfahren bezeichnet, die den Vergleich zweier Zeichensequenzen über eine visuelle Repräsentation wie in Abbildung 3.1(a) ermöglichen. Die Analyse der Visualisierung obliegt bei Dotplots im Allgemeinen einem Menschen, da dieser durch seine hohe visuelle Verarbeitungsleistung fähiger ist als eine Maschine.

Ursprünglich 1970 als Erleichterung zur Analyse der Ähnlichkeit von Gensequenzen [18] entwickelt, wurde das Visualisierungsverfahren namens "Dot-Matrix Plot" zunächst für diese Aufgabe optimiert und vorrangig dafür eingesetzt. Seine Verwendung im Bereich des Data-Minings und der Textanalyse erschließt sich aus dem artverwandten Prinzip, einen zeichenbasierten Vergleich zweier beliebiger Sequenzen zu veranschaulichen. Gensequenzen setzen sich in ihrer textlichen Repräsentation aus einem begrenzten Alphabet zusammen, durch das die einzelnen DNS-Basen repräsentiert werden. Die Adaption des Verfahrens auf Zeichenketten mit beliebiger Alphabetgröße [11], kann in der Abbildung 3.1(b) am Beispiel des Selbstvergleichs eines Quelltextes betrachtet werden. Diese Verwendungsform vergrößert die Auflösungsfähigkeit des Dotplots und unterstreicht seine universelle Nutzung zur Textanalyse, ebenso wie

zur Genanalyse in der Bioinformatik. Dotplots bilden die Zeichensequenzen orthogonal zueinander in einer zweidimensionalen Matrix ab. Die zu vergleichenden Sequenzen werden auf den Kanten der Matrix abgetragen, wobei jedes Zeichen einer Spalte und einer Zeile entspricht. Anschließend folgt das Verfahren einer einfachen Heuristik (siehe Algorithmus 3.1) und setzt in der Matrix einen Punkt, wenn das Zeichen der aktuellen Zeile verglichen mit dem Zeichen der aktuellen Spalte übereinstimmt. Aufgrund des zeichenweisen Vergleichs ist dieses naive Verfahren quadratisch in der Anzahl der Zeichen beider Sequenzen.

Algorithmus 3.1 Naiver Dotplot

Input: Sequenz s_1, s_2
Output: Dotplot-Matrix M_{dp}

```
t1 ← tokenize(s1)
t2 ← tokenize(s2)
{Initialisierung der Matrix M_dp}
M_dp ← [|t1|][|t2|]
for all x ∈ t1 do
  for all y ∈ t2 do
    if x = y then
      M_dp[x][y] ← 1
    end if
  end for
end for
return M_dp
```

Die partiell gefüllte Matrix eines Dotplots lässt sich direkt als Bild veranschaulichen und weist abhängig von den Eingangssequenzen unterschiedliche Muster auf. Die folgenden Beispiele (Abb. 3.2) entstammen den Selbstvergleichen künstlicher Zeichensequenzen zur Verdeutlichung der ausgeprägten Muster. Dotplots transportieren Informationen durch eine visuelle Sprache aus quadratischen und diagonalen Mustern [20]. Anhand der Muster lassen sich unterschiedliche Eigenschaften der einzelnen Sequenzen ableiten. Diese Muster ergeben sich aus der Natur des Verfahrens und der Repräsentation des Vergleichs. Quadrate kennzeichnen ungeordnete Bereiche sich wiederholender Zeichen, die bezüglich ihrer Position keiner Reihenfolge entsprechen. In der genauen Ansicht (Abb. 3.2(a)) wird durch ein Quadrat eine hohe Dichte übereinstimmender Zeichen eines gemeinsamen Alphabets gekennzeichnet [20]. Diagonalen stehen für identische Zeichen mit gleicher Reihenfolge innerhalb der zu vergleichenden Zeichensequenzen. Diese Muster beschreiben gerichtete Übereinstimmungen wie Kopien, Versionen oder Übersetzungen [20] in Bezug auf Zeichen und Wörter einer Sequenz. Abbildung 3.2(b) zeigt deutlich

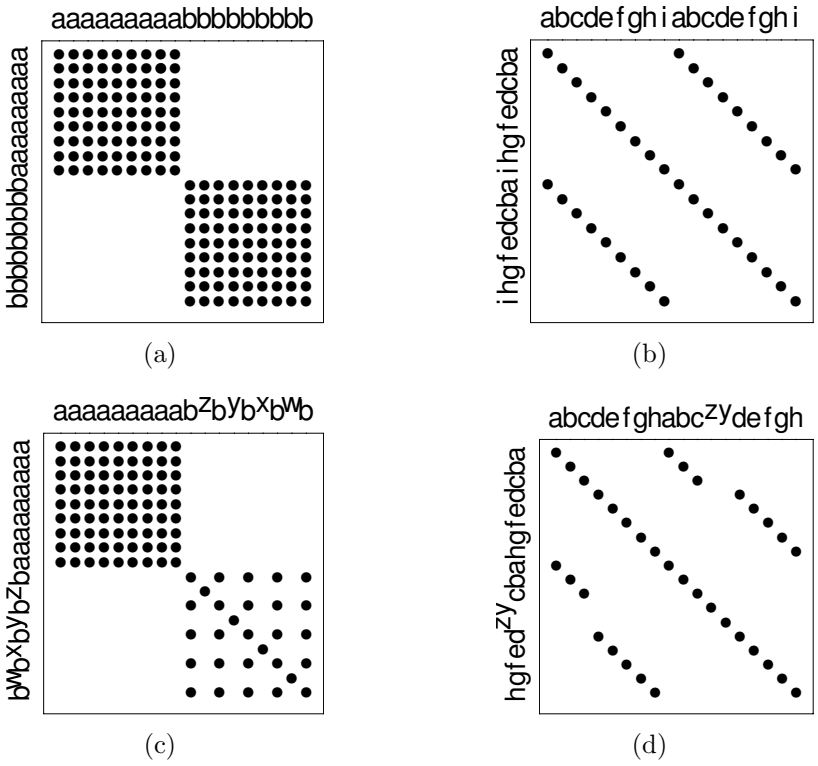


Abbildung 3.2. Quadrate und Diagonalen als Grundmuster der Dotplots [11]. Variationen innerhalb der Grundmuster markieren Transformationen innerhalb der Zeichensequenzen [20].

die identischen sich wiederholenden Zeichenabschnitte, die zu einer Doppelung der Hauptdiagonalen führen.

Beide Muster besitzen weitere Ausprägungen, die abhängig von den Transformationen der Eingabedaten zu verdichteten oder ausgedünnten Texturen führen können. In Abbildung 3.2(c) sind zwei verschieden dichte Quadrate sichtbar. Die ausgedünnte Variante entsteht durch das Einstreuen veränderter Zeichen innerhalb des homogenen Alphabets.

Eine abgeschwächte Form der vorangegangenen Zeichenstreuung ist das Einfügen veränderter Zeichenketten innerhalb identischer Sequenzabschnitte. Wie in Abbildung 3.2(d) leicht zu sehen ist, wurden innerhalb der zu vergleichenden Sequenzen kleine, nicht übereinstimmende Teile eingefügt. Parallel verschobene Nebendiagonalen kennzeichnen diese Veränderung und der ersichtliche Versatz zur Hauptdiagonalen drückt unmissverständlich die Zeichenentfernung der Veränderung von der Originalstelle aus.

Die durch Dotplots visualisierten Eigenschaften der Zeichensequenzen rei-

chen damit von einfachen Kopien, über Einfügungen bzw. Löschungen, bis hin zu Umordnungen, Einstreuungen und anderen Modifikationen. Die Komplexität eines Dotplots hängt folglich direkt mit der Art und Weise der Text- und Zeichentransformationen zusammen. Jedoch lässt sich jeder noch so komplexe Dotplot durch die enthaltenen Quadrate und Diagonalen eindeutig interpretieren [20].

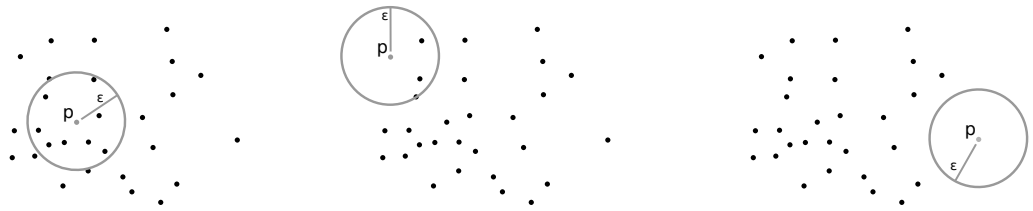
Der Selbstvergleich langer Gensequenzen und allgemeiner Zeichensequenzen steht beim Dotplot-Verfahren im Vordergrund. Obwohl es ursprünglich für den Vergleich gleichlanger Sequenzen gedacht war, eignet es sich ebenfalls für Vergleiche beliebig langer und insbesondere verschiedener Zeichensequenzen. Diese Variante setzen wir in unserer Entwicklung ein, da wir generell paarweise Vergleiche verschiedener Eingangssequenzen durchführen. Die Verallgemeinerung der dadurch entstehenden Muster sind Rechtecke und Geraden innerhalb zweidimensionaler Plots. Auf diese Erscheinungen werden wir später erneut eingehen.

3.1.2 Automatische Clusteranalyse mit DBScan

DBScan ist ein Algorithmus zur Identifikation von Klassen innerhalb zwei- und mehrdimensionaler Daten [14]. Der Clusteralgorithmus zählt zu den Verfahren der explorativen Datenanalyse. Er basiert auf dem Konzept der unüberwachten Aufteilung einer Menge unbekannter Datenobjekte, in eine abzählbare Menge verschiedener Gruppierungen, abhängig von den Eigenschaften der Datenobjekte [44]. Das Verfahren gruppiert die Datenobjekte einer Datenmenge, abhängig von ihrer räumlichen Verortung, in Bereiche hoher Dichte. Die Datenwerte jedes Datenobjekts entsprechen bei räumlichen Strukturen den Koordinaten des Objekts, welches dadurch als Datenpunkt repräsentiert wird. DBScan arbeitet nach dem Prinzip der Dichteschätzung von Datenpunktbereichen zur Identifikation einer beliebigen Clusteranzahl [39]. Das Verfahren folgt der Methodik, jeden Datenpunkt einem Cluster zuzuordnen, wenn dieser von einer Mindestanzahl benachbarter Datenpunkte eingeschlossen ist.

Der Clusteralgorithmus unterscheidet zwischen drei Fällen eines Datenpunktes [14, 37]. Kernpunkte (*core points*) sind von einer festgelegten Anzahl anderer Datenpunkte umringt (Abb. 3.3(a)). Rauschpunkte (*noise points*) besitzen gar keine Nachbarn (Abb. 3.3(c)). Datenpunkte abseits der anderen Kriterien sind Randpunkte (*border points*), wie in Abbildung 3.3(b) zu sehen ist.

Für die Überprüfung dieser Kriterien ist zunächst die Definition eines Abstandes $dist(p, q)$ zwischen zwei Datenpunkten p und q aus der Datenmenge D notwendig. Welche konkrete Distanzfunktion zum Einsatz kommt, ist im Vorhinein entsprechend zu definieren. In unserem Fall wird die euklidische Distanz für die Datenpunkte in einem zweidimensionalen Raum berechnet.



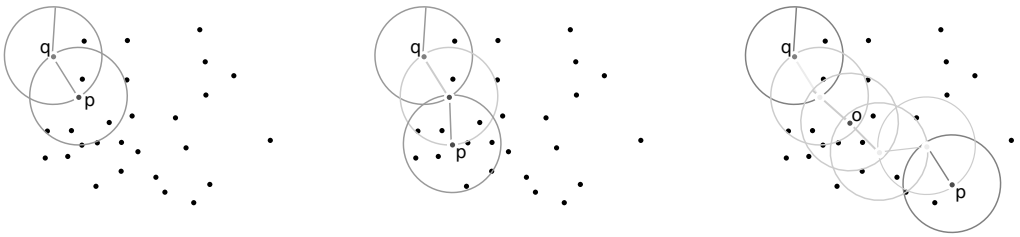
(a) Kernpunkt (b) Randpunkt (c) Rauschen

Abbildung 3.3. Varianten der Datenpunkte des DBScan-Algorithmus.

Zur Berechnung der ϵ-Nachbarschaft N_ϵ eines Datenpunktes wird ein kreisrunder Bereich mit festem Radius ϵ um den Datenpunkt definiert. N_ϵ ist für einen Datenpunkt p als die Menge (3.1) definiert, die alle von p verschiedenen Punkte q enthält, deren Abstand zu p kleiner ist als ϵ.

$$N_\epsilon(p) \quad = \quad \{\ q \mid dist(p,q) < \epsilon,\ p \neq q,\ p \in D,\ q \in D\ \} \tag{3.1}$$

Unter der Bedingung $|N_\epsilon(p)| \geq MinPts$ wird p als Kernpunkt des Clusters definiert. Der Parameter $MinPts$ definiert die Mindestanzahl benachbarter Punkte eines Kernpunktes.



(a) Direkt dichteerreichbar (b) Dichteerreichbar (c) Dichteverbunden

Abbildung 3.4. Kriterien für die Dichtemessung im DBScan-Algorithmus. Hier mit den Datenpunkten o, p, q und der zugehörigen ϵ-Nachbarschaft.

Wenn p ein Kernpunkt ist und sich der Datenpunkt q innerhalb seiner ϵ-Nachbarschaft befindet, $q \in N_\epsilon(p)$, so ist q von p aus *direkt dichteerreichbar* (Abb. 3.4(a)). Zwischen zwei Kernpunkten ist diese Beziehung symmetrisch, handelt es sich bei einem der Punkte um einen Randpunkt ist das nicht der Fall [14].

Tritt diese Bedingung nicht ein und existiert jedoch zwischen den Punkten p und q eine Menge anderer Datenpunkte, die untereinander direkt dichteerreichbar sind, dann sind p und q *dichteerreichbar* (Abb. 3.4(b)).

Existiert ein Datenpunkt n, der von keinem Kernpunkt aus dichteerreichbar ist und selbst das Kriterium für einen Kernpunkt nicht erfüllt, so wird dieser als Rauschen bzw. keinem Cluster zugehörig definiert. In jedem anderen Fall handelt es sich um einen Randpunkt, der definitionsgemäß weniger andere Datenpunkte als $MinPts$ in seiner ϵ-Nachbarschaft besitzt, jedoch von einem Kernpunkt aus dichteerreichbar ist.

Die Dichteerreichbarkeit ist für Kernpunkte eine explizit symmetrische Beziehung, weil als Vorbedingung die direkte Dichteerreichbarkeit innerhalb einer Punktemenge vorausgesetzt wird. Diese Beziehung ist transitiv für alle übrigen Datenpunkte [14] außer dem Rauschen. Um die symmetrische Beziehung auf Randpunkte auszudehnen, wird das Kriterium der Dichteverbundenheit definiert. Zwei nicht dichteerreichbare Randpunkte $p \notin N_\epsilon(q)$ und $q \notin N_\epsilon(p)$ sind *dichteverbunden*, sobald ein Kernpunkt o mit $|N_\epsilon(o)| \geq MinPts$ existiert, von dem aus beide Randpunkte, $p, q \in N_\epsilon(o)$, dichteerreichbar sind (Abb. 3.4(c)). Dieses Kriterium definiert eine symmetrische Beziehung für alle Datenpunkte außer dem Rauschen [14].

Die genannten Kriterien sind die Grundlage für die Klassenzuordnung der einzelnen Datenpunkte innerhalb der Datenmenge. Die Clusteranalyse beginnt mit dem Zustand, dass jeder Datenpunkt ein eigenes Cluster bildet. Alle Datenpunkte werden auf die einzelnen Datenpunktvarianten getestet. Jeder Datenpunkt, der innerhalb der ϵ-Nachbarschaft eines Kernpunktes liegt, wird dem Cluster des Kernpunktes zugeordnet. Diese Relation wird auf alle Datenpunkte der Datenmenge propagiert und der Algorithmus terminiert, wenn alle Datenpunkte einem Cluster zugeordnet sind oder als Rauschen markiert wurden [39].

Mit einer theoretischen Laufzeit von $\mathcal{O}(n(log\ n))$ in Abhängigkeit der Anzahl der Datenpunkte n, ist die richtige Wahl der unterlegten Datenstruktur für DBScan aus Gründen der Effizienz wichtig [44]. Angesichts der Datenmengen, wie sie in unserem Ansatz zum Tragen kommen, hat das erhebliche Auswirkungen auf den Zeitfaktor der Textanalyse. Aus diesem Grund nutzen wir einen kD-Tree als spezialisierte Datenstruktur für DBScan.

Ein kD-Tree [9] ist ein räumlicher Index, der seine k-dimensionalen Daten durch effiziente Partitionierung eines mehrdimensionalen Raumes in einer Baumstruktur ablegt. Jedes Datenelement wird als Knoten im Baum unter einer eindeutigen Koordinatenkombination abgelegt, die sich, wie in unserem Fall, direkt aus den Datenwerten ergibt. Der kD-Tree zählt zu den Datenstrukturen für Bereichssuchen und räumliche Indizierung. Er ist in diesem Sinne verwandt mit dem R-Tree, der im Gegensatz zum kD-Tree seine Daten nicht unter einer Koordinate in einem partitionierten Raum ablegt, sondern die Daten einem entsprechenden Bereich zuweist. Je nach Art und Weise der implementierten Schnittstellen sind beide Datenstrukturen für DBScan geeignet.

3.2 Text-Reuse-Extraktion

In diesem Abschnitt werden die bereits vorgestellten Werkzeuge kombiniert und unser Algorithmus zur Identifikation und Extraktion von Text-Reuse-Instanzen vorgestellt. Angefangen mit dem Datenmodell, über die notwendigen Stufen der Datenvorverarbeitung, bis hin zur Analyse der Datenrepräsentationen werden wir unseren Ansatz für dieses Textanalyseproblem betrachten und diskutieren.

3.2.1 Modellierung

Unser Information-Retrieval-System hat die Aufgabe, wiederverwendete Textabschnitte innerhalb einer Menge von Textabschnitten aus zwei Eingabedokumenten zu erkennen und zu extrahieren. Dabei versuchen wir, durch einen Vergleich syntaktisch ähnlicher Textfragmente inhaltlich wiederverwendete Textabschnitte zu identifizieren. Abbildung 3.5 zeigt eine schematische Aufteilung der einzelnen Schritte unserer Entwicklung.

Ein Textdokument wird bei unserem Verfahren als eine Liste von Wort-N-Grammen mit ihren zugehörigen Positionen im Textdokument repräsentiert. Die Textdokumente werden in Fragmente definierter Länge N zerlegt. Es wird ein Index erzeugt, dessen Schlüssel aus den extrahierten Wort-N-Grammen bestehen. Jedes Wort-N-Gramm verweist auf seine zugehörige Positionsliste.

Nach einheitlicher Repräsentation der Quelldokumente bestehen Query und Datenbasis, in unserem Fall, aus den Wort-N-Grammen beider Textdokumente. Die Aufgabe besteht darin, die Abschnitte beider Dokumente zu identifizieren, die ähnliche Textfragmente enthalten.

Wir verwenden während des Retrievals wiederverwendeter Textabschnitte ein mehrstufiges Relevanzkriterium. Die extrahierten Wort-N-Gramme werden in einer ersten Stufe durch eine boolesche Relevanzfunktion als identisch oder nicht identisch markiert. Anschließend folgt eine Permutation der Positionen aller gemeinsamen identischen Wort-N-Gramme. Diese Positionstupel werden in einem zweidimensionalen Raum als Punkte interpretiert. Eine zweite Stufe analysiert dichte Bereiche in dem entstehenden Punktefeld. Die Punkte in diesen Bereichen werden zu Gruppen zusammengefasst.

Unser Relevanzkriterium setzt sich damit aus einer zeichenbasierten Identitätsprüfung im ersten Schritt und einer Clusteranalyse im zweiten Schritt zusammen. Über die in den Gruppen enthaltenen Wort-N-Gramme können die Positionen ähnlicher Textstellen in beiden Dokumenten angegeben werden.

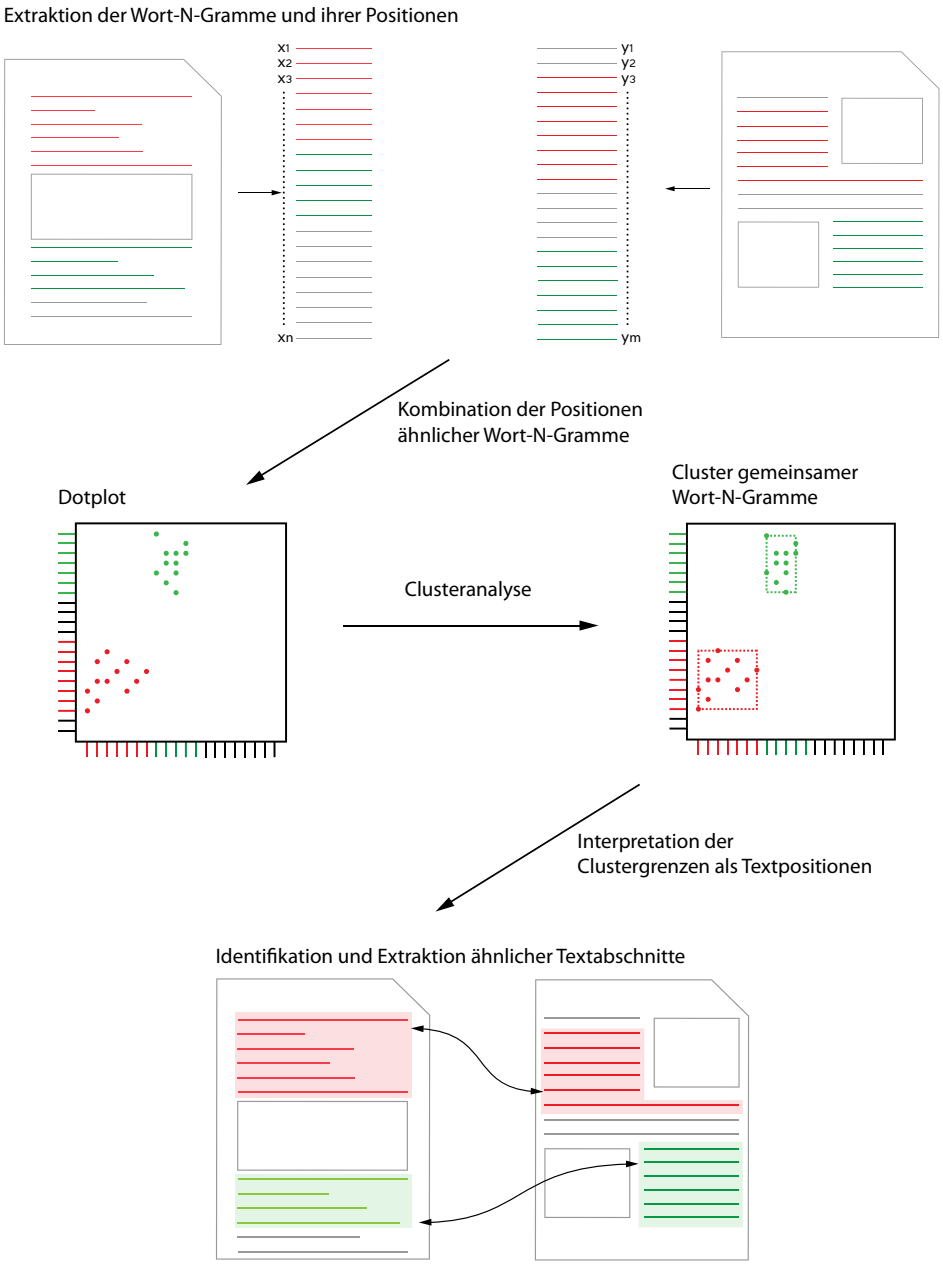


Abbildung 3.5. Verarbeitungsschema der Textanalyse zur Text-Reuse-Erkennung und Text-Reuse-Extraktion.

Der Prozessaufbau orientiert sich an der Verarbeitungs-Pipeline für IR-Systeme zur Identifikation von Plagiaten [8, 41]. Er besteht aus Datenakquise- und vorverarbeitung, Datenanalyse und Nachverarbeitung. Diese Schritte werden wir nun ausgehend von der Eingabe zweier Textdokumente detaillierter betrachten.

3.2.2 Textvorverarbeitung

Die Textvorverarbeitung gliedert sich in mehrere Einzelschritte, die jedes Textdokument für die Analyse normalisieren. Wir setzen für diese Normalisierung eine Pipeline-Architektur zur Definition und Anwendung unserer Textfilter ein. Die Filter-Pipeline behandelt ein eingehendes Textdokument als reine Zeichensequenz und wendet nach unserer Definition die Textfilter in den nun beschriebenen Schritten an. Am Beispiel des folgenden Blindtextes ist die Veränderung in jeder Stufe zu sehen.

> Far far away, behind the word mountains, far from the countries Vokalia and Consonantia, there live the blind texts.

Tokenisierung

> (far, far, away, behind, the, word, mountains, far, from, the, countries, vokalia, and, consonantia, there, live, the, blind, texts)

Die Tokenisierung extrahiert aus der eingehenden Zeichensequenz alle erkannten Wörter. Unter Vernachlässigung der Satzinterpunktionen sowie der Groß- und Kleinschreibung besteht die Ausgabe aus vereinheitlicht kleingeschriebenen Wörtern.

Stoppwortfilter

> (word, mountains, countries, vokalia, consonantia, live, blind, texts)

Der Stoppwortfilter entfernt alle Wörter, die in einer sprachabhängigen Stoppwortliste verzeichnet sind.

Synonymfilter

> (word, mountain, country, vokalia, consonantia, live, blind, text)

Ein Synonymfilter vereinheitlicht alle Vorkommen eines Wortes auf eine festgelegte Alternativform. Die Möglichkeiten zur Normalisierung sehen eine Ersetzung durch das häufigste oder seltenste Synonym vor. Informationen über die Synonyme eines Wortes erhalten wir durch die Benutzung von Synonymdatenbanken. Für die englische Sprache wird WordNet [32] genutzt.

Stemming-Filter

> (word, mount, count, vokal, conso, live, blind, text)

Von jedem Wort des aktuellen Datenstroms wird die Stammform gebildet. Das verwendete Suffix-Stemming [15] kürzt jedes Wort auf eine festgelegte Präfixlänge. Die verbleibenden Wortfragmente können, abhängig von dem definierten Ausmaß der Kürzung, sowohl Stammformen als auch stark verkürzte Wortanfänge beinhalten. In beiden Fällen werden durch das Stemming Flexionen und unter Umständen auch typografische Fehler ausgeglichen.

Chunking-Filter

> {
> (word, mount, count, vokal, conso),
> (mount, count, vokal, conso, live),
> (count, vokal, conso, live, blind),
> (vokal, conso, live, blind, text)
> }

Der Chunking-Filter verarbeitet den Datenstrom Wort für Wort und erzeugt bei jedem Wortwechsel ein neues Wort-N-Gramm. Als einziger Parameter wird die Anzahl N der Wörter im Wort-N-Gramm festgelegt. Um auch hier die Variabilität der Textfragmente zu berücksichtigen, müssen alle auftauchenden Wortkonstruktionen mit ihrer Position erfasst werden. Aus diesem Grund wird der Filter wortweise angewandt. Das Einfügen und Verändern

einzelner Wörter innerhalb einer Sequenz von Wort-N-Grammen drückt sich, durch die wortweise Extraktion des Textes, im Dotplot direkt als Verschiebung zur Hauptdiagonalen aus.

Wort-N-Gramm-Sortierung

```
{
    (conso, count, mount, vokal, word),
    (conso, count, live, mount, vokal),
    (blind, conso, count, live, vokal),
    (blind, conso, live, text, vokal)
}
```

Als letzte Stufe der Textvorverarbeitung wird eine alphabetische Sortierung der Wörter jedes Wort-N-Gramms vorgenommen. Dieser Filter normalisiert die Daten unter der Annahme, dass wiederverwendete Textabschnitte ähnliche oder gleiche Wörter in Bezug auf das Original beinhalten, um die Semantik des Inhalts zu erhalten. Umgestellte Phrasen und Satzfragmente innerhalb der Textfragmente werden so auf gleiche Formen zurückgeführt. Damit erlaubt dieser Filter eine Textanalyse auf wortähnlichen Textabschnitten, die durch diese Normalisierung erst vergleichbar werden. Die Folge ist eine erhöhte Variabilität der Eingangstexte durch die Abdeckung dieser Texttransformationen.

Dokumentrepräsentation nach der Vorverarbeitung

Die Ausgabe der Textvorverarbeitung ist eine Liste von Textobjekten. Jedes Textobjekt besteht aus der Position des ursprünglichen Wort-N-Gramms im originalen Textdokument und dem normalisierten Wort-N-Gramm. Damit lassen sich die stark verfremdeten Wort-N-Gramme jederzeit auf ihre ursprünglichen Textstellen zurückführen. Die Menge W vereint alle aus der Vorverarbeitung extrahierten normalisierten Wort-N-Gramme. Jedes normalisierten Wort-N-Gramme wird künftig als Token t bezeichnet.

Die bereits angesprochene Dokumentrepräsentation wird über die extrahierten Textobjekte aufgebaut. Während der Normalisierung beider Dokumente werden die Positionen der extrahierten Wort-N-Gramme innerhalb der Dokumente durch die Funktion *pos* in einer Liste aufgezeichnet. Jedes Dokument d sei eine Menge von Wörtern. Eine Dokumentrepräsentation $\mathbf{d} : W \to \mathcal{P}(\mathbb{N})$ von d, ist die Abbildung aller Token in W auf eine Menge von Positionen innerhalb des Dokuments d. Jedes der beiden Dokumente d_1 und d_2 für die

Textanalyse, wird als Index mit

$$\mathbf{d}_1 = \{\ (t, pos(t, d_1)) \mid t \in W\ \}$$

und

$$\mathbf{d}_2 = \{\ (t, pos(t, d_2)) \mid t \in W\ \}$$

repräsentiert. Die beiden Dokumentrepräsentationen $\mathbf{d}_1$ und $\mathbf{d}_2$ sind assoziative Datenstrukturen (*maps*), mit eindeutigen Schlüssel-Wert-Paaren. Jedes Token ist eindeutig in der Schlüsselmenge von $\mathbf{d}_1$ und $\mathbf{d}_2$ identifizierbar und verweist auf die Positionsliste im Originaldokument.

3.2.3 Analyse und Extraktion

Die Analyse ist, genau wie die Textvorverarbeitung, in mehrere Stufen unterteilt. Ausgehend von der normalisierten Dokumentzerlegung, leiten die nächsten Schritte eine konkrete Analyse ein. Auf Basis eines Dotplots wird der direkte Vergleich beider Dokumente abgebildet und dient als Grundlage für die Clusteranalyse mit DBScan.

Identifikation gemeinsamer Token

Die Analyse beginnt mit der Berechnung einer Schnittmenge beider Indexschlüsselmengen. Diese Tokenmenge $\mathbf{T} = \mathbf{d}_1 \cap \mathbf{d}_2$ ist das Ergebnis einer ersten Relevanzbestimmung durch eine boolesche Relevanzfunktion. Mit den verbleibenden gemeinsamen Indexschlüsseln in $\mathbf{T}$ können fortan die gemeinsamen Textfragmente direkt referenziert werden. Für jedes Token $t \in \mathbf{T}$ wird auf die Positionslisten der beiden Dokumente $\mathbf{d}_1$ und $\mathbf{d}_2$ zugegriffen. Die erhaltenen Positionslisten werden über das kartesische Produkt miteinander kombiniert. Daraus resultiert das Punktefeld

$$\mathbf{F} = \{\ (x, y) \mid x \in \mathbf{d}_1(t), y \in \mathbf{d}_2(t), t \in \mathbf{T}\ \}$$

welches alle Positionen gemeinsamer Token enthält. Es beinhaltet eine Menge von Datenpunkten, die die gemeinsamen Token beider Dokumente repräsentieren. Jeder Datenpunkt besitzt mit der x-Koordinate eine Referenz auf das erste Dokument d_1 und mit der y-Koordinate auf das zweite Dokument d_2. Eine Visualisierung dieses Punktefeldes ist der Dotplot. Beide Dokumentrepräsentationen werden entlang der beiden Achsen abgetragen (Abb. 3.6). Der Ursprung bei unserer Variante des Dotplots befindet sich in der linken unteren Ecke und markiert den Start beider Dokumente. Die Skalengröße entspricht dabei der Länge jedes Dokuments. Entlang der Skala ist die absolute Zeichenposition abgetragen. Eine Menge gemeinsamer Token wird als Punktewolke visualisiert.

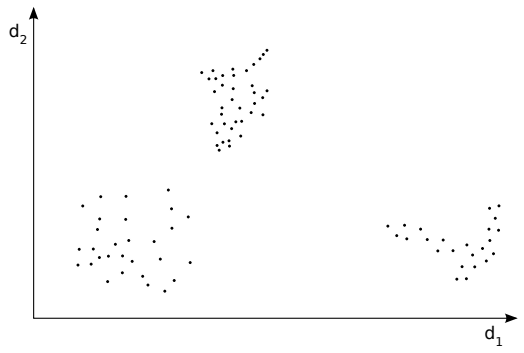


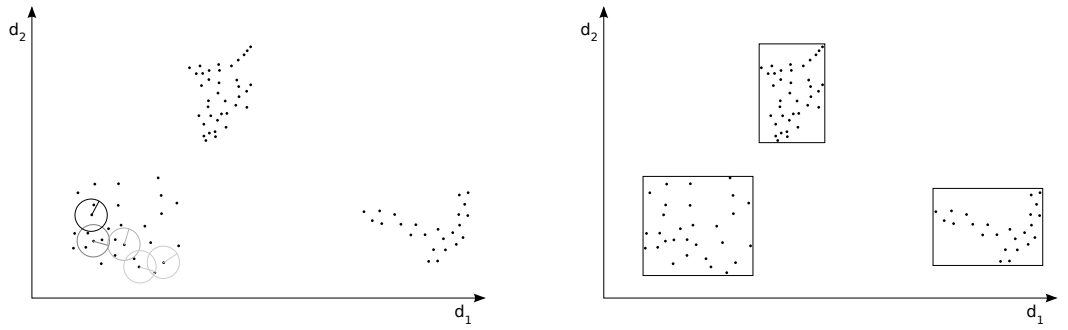
Abbildung 3.6. Punktefeld der gemeinsamen Token zweier Dokumente.

Alle Positionen sind durch die vorangegangene Kombination bekannt und werden nun unverändert weiterverwendet. Die nächste Analysestufe erhält das Punktefeld **F** als Grundlage seiner Berechnungen.

Hier setzen wir ein Clusteranalyseverfahren zur Unterteilung des Punktefeldes ein. In der Visualisierung der Dotplots erwarten wir, bei vielen übereinstimmenden Wort-N-Grammen, partielle Häufungen innerhalb des Punktefeldes und vereinzelt abseits liegende Punkte zu sehen. Das Gros der Punkte sollte sich innerhalb dichter Bereiche sammeln und dadurch Textbereiche mit vielen gleichen Textfragmenten abbilden. Es wird angenommen, dass aufgrund des zweidimensionalen Punktefeldes und der erwarteten Punktewolken ein dichtebasiertes Clusteranalyseverfahren die Aufgabe zufriedenstellend löst. Uns standen die beiden Varianten DBScan und MajorClust [40] zur Verfügung.

Als parameterloses Verfahren wurde zunächst MajorClust in Betracht gezogen. Die Abbildung der Daten im Dotplot und die Handhabung der auftretenden Muster war jedoch ungeeignet. Die erwartete Streuung der Punkte und die inselartigen Häufungen widersprachen dem Konzept hinter MajorClust, jeden Punkt konsequent einem Cluster zuzuordnen. Weit entfernte vereinzelte Punkte, die für einen menschlichen Betrachter als Ausreißer galten, wurden durch den Algorithmus zwangsläufig einem Cluster zugeordnet. Mit Hinblick auf die Lösungsmenge ergaben sich Ballungseffekte, die eine Differenzierung von kleinen Häufungen und offensichtlichen Rauschpunkten nicht zuließen.

DBScan besitzt das Konzept der Rauschpunkte und ist darüber hinaus besser für die Verwendung niedrig-dimensionaler Datenwerte geeignet [37]. Der Einsatz des Cluster-Algorithmus in Verbindung mit einem kD-Tree in unserer Clusteranalyse, erwies sich als vielversprechende Kombination. Alle gesammelten Rauschpunkte können dadurch getrennt von den Datenpunkten der entstandenen Cluster verarbeitet werden. Die Erkennung der Dotplot-Muster innerhalb stark verrauschter Punktefelder ist ebenfalls möglich. Für den Einsatz



(a) DBScan-Instanz der ersten Clusteranalyse auf den abstrahierten gemeinsamen Tokenpunkten.

(b) Ergebnis der ersten DBScan-Instanz sind die lokalen Cluster sehr dichter Punktebereiche.

Abbildung 3.7. Clustering der Datenpunkte innerhalb lokaler Dichtebereiche.

des DBScan-Algorithmus mussten wir die notwendigen Parameterabhängigkeiten, die spezifisch für die Daten einzustellen sind, in Kauf nehmen.

Clustering lokaler Dichtebereiche

Die Datenpunkte in **F** sind nach dem Direktvergleich noch unbrauchbar für die Clusteranalyse. Für ein schnelles Clustering durch DBScan werden sie in einen zweidimensionalen kD-Tree überführt, einen 2D-Tree. Dieser Baum wird einer ersten DBScan-Instanz zur unüberwachten Klassifizierung der Punktemenge übergeben. DBScan berechnet mit den beiden Parametern für den Radius ϵ und das Kernpunktkriterium $MinPts$ ein Clustering der Datenpunkte (Abb. 3.7(a)). Die berechneten Cluster schließen Datenpunkte ein, die sich innerhalb dichter Punktebereiche befinden (Abb. 3.7(b)).

Aufgrund der Datenrepräsentation und der durch den Textvergleich entstehenden Muster besitzen die Cluster keine freie Form. Sie werden als rechteckige Strukturen innerhalb der Analyse, wie auch der Visualisierung behandelt. Der linke untere Eckpunkt repräsentiert nach diesem Modell die Startposition der gemeinsamen Textstelle in jedem der beiden Dokumente. Demzufolge ist der obere rechte Eckpunkt eine direkte visuelle Abbildung auf die Endposition der Textstelle.

Jedes Cluster wird nach der Clusteranalyse auf seine Ausmaße in beide Dimensionen überprüft. Diese Kontrolle bezieht sich auf die Anzahl der eingeschlossenen Textzeichen bezüglich beider Dokumente. Über einen Filter werden alle Cluster verworfen, deren Ausmaße einen festgesetzten Schwellwert in einer der beiden Dimensionen unterschreiten. Wir haben den Schwellwert auf 90 Zeichen festgelegt, welcher dadurch eine Mindestlänge jedes Clusters von etwa zwei Sätzen vorschreibt.

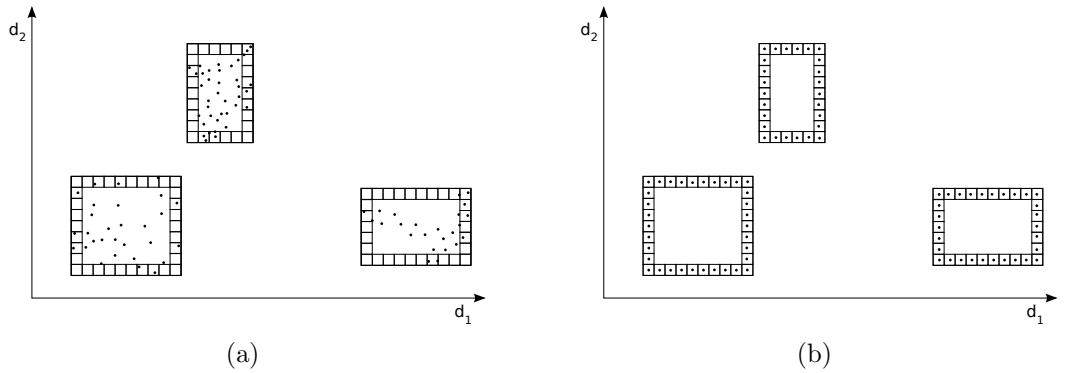
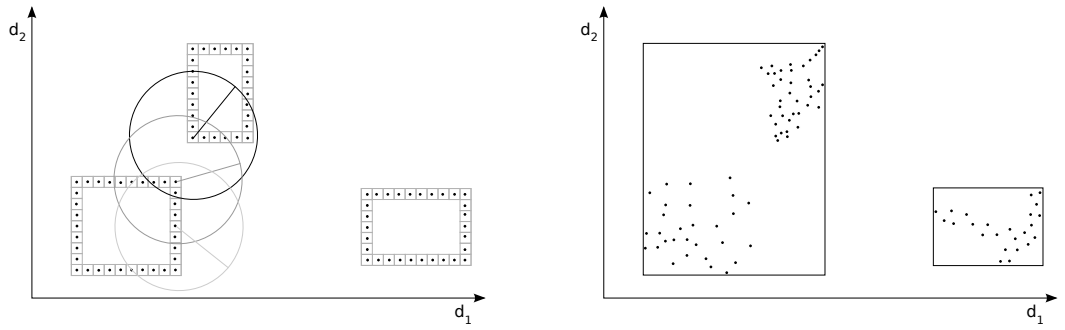


Abbildung 3.8. Künstliche Fragmentierung der lokalen Clusterränder.

Alle nicht gefilterten Cluster werden anschließend auf Überschneidungen mit anderen Clustern hin überprüft. Weiterhin testet unser Verfahren jedes Cluster auf das Enthaltensein innerhalb der übrigen Cluster dieses Clusterings. Dadurch wird Redundanz vermieden und gegebenenfalls bereinigt.

Danach werden die verbleibenden Cluster fragmentiert. Unser Verfahren zerlegt jedes Cluster entlang seines Randes in homogene Bereiche fester Ausdehnung (Abb. 3.8(a)). Von jedem dieser Bereiche wird der geometrische Mittelpunkt berechnet (Abb. 3.8(b)) und in einem weiteren kD-Tree abgelegt. Die Mittelpunkte der Fragmente repräsentieren nun die originalen räumlichen Datenausmaße. Die vorher gruppierten Tokenpunkte werden nun durch einen Bereich mit homogener Zerlegung am Rand abstrahiert. Dieses Vorgehen gewährleistet die Möglichkeit, mit der nächsten Clustering-Instanz über die erkannten lokalen Cluster hinaus in benachbarte Bereiche zu blicken. Homogene Teilbereiche sichern die Wiedererkennung vorhandener Cluster und ermöglichen die Erkennung neuer Verbindungen zwischen den Clustern.

Das Clustering dieser DBScan-Instanz wird durch die eingesetzten Teilschritte und Filter darauf hin optimiert, sehr dichte Bereiche in kleinen Regionen zu repräsentieren. Ähnlich dem lokalen Sequence-Alignment bilden sich hier Inseln dichter Bereiche aus. Auch wenn die Mindestgröße für ein Cluster im Text auf etwa zwei Satzlängen festgelegt ist, erwecken die erkannten Regionen den Eindruck der Zusammengehörigkeit, obwohl sie mehrere Absätze im Text auseinanderliegen können. Ein Mensch würde derartig getrennte, ähnliche Abschnitte im Großteil der Fälle zu einer zusammenhängenden Text-Reuse-Instanz zählen. Um diesem Umstand Rechnung zu tragen, verarbeitet eine weitere Clusteranalyse die fragmentierten Cluster in einer größer bemessenen Umgebung.



(a) DBScan-Instanz der zweiten Clusteranalyse mit den künstlich erzeugten Fragmentierungen als Grundlage.

(b) Finale Cluster der zweiten DBScan-Instanz vereinen alle lokal naheliegende Cluster.

Abbildung 3.9. Clustering der Mittelpunkte aller homogenen Teilfragmente der ersten Clusteranalyse.

Clustering globaler Dichtebereiche

Diese DBScan-Instanz verwendet eine größere ϵ-Nachbarschaft. Dieses Vorgehen erlaubt es, auch in weiter entfernte Umgebungen rund um die Cluster zu blicken. Das Prinzip hinter diesem Clustering ist dem für lokale Dichtebereiche nachempfunden. Die Fragmentierung dient hier als Datengrundlage, die originalen Datenpunkte spielen keine Rolle mehr. Der Clusteralgorithmus verfährt entlang der geometrischen Mittelpunkte jedes Fragments und gruppiert naheliegende Fragmente (Abb. 3.9(a)). Durch die vergrößerte ϵ-Nachbarschaft wird eine Maximierung zusammengehöriger Bereiche gefördert (Abb. 3.9(b)). Die Umkreissuche entlang der Ränder ermöglicht dadurch eine Zusammenfassung weiter entfernter Textfragmente.

Wie in der vorangegangenen Clusteranalyse wird das Clustering durch die zusätzlichen Filterverfahren überprüft und die einzelnen Cluster gegebenenfalls korrigiert. Eine Filterung nach der Clustergröße findet nach dieser Clusteranalyse nicht mehr statt, da die erkannten Gruppierungen bereits über die Mindestlänge hinaus angewachsen sind.

Die gesamte Anwendung dieser Clusteranalyse dient der verbesserten Gruppierung ähnlicher Textbereiche unter Berücksichtigung mehr oder minder starker Textveränderungen. Diese Veränderungen wirken sich als Verschiebungen der Haupt- und Nebendiagonalen innerhalb eines Dotplots aus. Sie führen dadurch zu Aussparungen zwischen den ähnlichen Textbereichen, die durch diese Clusteranalyse zu einem gewissen Grad kompensiert werden können. Durch die Maximierung der Übereinstimmungen ähnlicher Textabschnitt innerhalb der Cluster, sehen wir hier eindeutige Parallelen zum Konzept des globalen Sequence-Alignment.

3.2.4 Nachverarbeitung

Die finale Ausgabe des Analyseschrittes ist ein Clustering bzw. eine Liste von Datenpunktpaaren, die jeweils ein Cluster diagonal aufspannen. Während der Nachverarbeitung wird diese Liste linear durchlaufen und die Datenpunktpaare direkt auf die beiden Textdokumente angewendet. Jedes Listenelement repräsentiert ein rechteckiges Cluster mit zwei diagonalen Aufhängepunkten. Der erste Datenpunkt markiert mit der linken unteren Ecke die Anfangsposition, der zweite Datenpunkt markiert mit der rechten oberen Ecke die Endposition des gemeinsamen Textabschnittes innerhalb beider Textdokumente.

Die extrahierten Textabschnitte beinhalten identische Textbestandteile im Sinne identischer Token. Sie schließen durch die Clusteranalyse der gemeinsamen Token allerdings auch die häufigen Texttransformationen ein, die durch das Kopieren, Einfügen, Umstellen und Paraphrasieren entstehen. Damit werden ebenfalls ähnliche Textbestandteile abgedeckt. Extrahierte Textabschnitte sind demnach identifizierte Text-Reuse-Instanzen. Weitere Verarbeitungsschritte werden von anderen IR-Systemen übernommen, die erkannte Text-Reuse-Instanzen verwenden und visualisieren.

3.2.5 Verwandte Verfahren

Verfahren zur Erkennung wiederverwendeter Texte wurden bereits im Rahmen der Entwicklung von Plagiatdetektoren hervorgebracht [30, 28, 42]. Von besonderem Interesse sind hier die Entwicklungen speziell für den Retrieval-Task der *externen Analyse* zweier Dokumente. Darunter wird die paarweise Textanalyse eines verdächtigen Dokumentes zusammen mit einem möglichen Quelldokument verstanden. Die Aufgabe besteht in der Erkennung gemeinsamer ähnlicher Texte zur Verifizierung des Quelldokuments und zur Identifizierung plagiierter Stellen.

Zum aktuellen Zeitpunkt ist ein weiteres Verfahren aus diesem Bereich bekannt, welches unter Einsatz von Dotplots eine Textanalyse zur Plagiaterkennung durchführt. Encoplot [19] basiert auf einer reduzierten Variante des Dotplot-Verfahrens und setzt vorrangig auf die Erkennung zusammenhängender diagonaler Muster. Jedes Dokument wird bei diesem Verfahren als Menge von Zeichen angesehen. Zunächst werden die Dokumente in fixe Zeichenketten mit jeweils 16 Zeichen zerlegt, dabei wird jede Position in einer Liste aufgezeichnet. Für identische Zeichenketten aus beiden Dokumenten werden ihre aufsteigend sortierten Positionslisten beginnend mit dem kleinsten Wert miteinander kombiniert. Das Verfahren fasst damit die erste Stelle der gemeinsamen Zeichensequenz in einem Dokument mit der jeweils ersten Stelle in einem anderen Dokument zusammen. Dieses Vorgehen wird für alle Positionen in

den Listen wiederholt. Der entstehende Dotplot weist damit neben vereinzelten Punkten überwiegend Diagonalen auf und bildet übereinstimmende Sequenzen nur einmal ab. Die weiterführende Verarbeitung setzt Optimierungsverfahren zur Erkennung diagonaler Muster ein. Alle gefundenen Diagonalen werden gruppiert. Das gesamte Vorgehen soll eine lineare Laufzeit in Bezug auf den Vergleich der Dokumente gewährleisten.

Durch die Priorisierung diagonaler Muster werden implizit alle weiteren Vorkommen gemeinsamer Zeichensequenzen innerhalb der Dokumente durch Encoplot ignoriert. Weiterhin wird durch den Einsatz eines schmalen Toleranzbereichs um die diagonalen Ausprägungen, die Möglichkeit eingeschränkt, auch stark eingestreute Textveränderungen zuverlässig als zusammenhängend zu erkennen.

Unser Verfahren besitzt gegenüber dem Encoplot lediglich Ähnlichkeiten in Bezug auf die Erstellung der Dotplots. Es orientiert sich sowohl an diagonalen als auch an quadratischen und rechteckigen Mustern im Dotplot. Jeder Dotplot wird als vollständiges Punktefeld mit allen Kombinationen gemeinsamer Sequenzen aufgebaut. Die detaillierte Vorverarbeitung und die eingesetzten Heuristiken zwischen den beiden Clusteranalysen sind ausschlaggebend für die Unterschiede zum Encoplot. Auch beim Encoplot existiert ein Verfahren zur Gruppierung der in Nachbarschaft befindlichen Punkte, welches durch weitere Heuristiken gestützt wird. Im Gegensatz dazu setzen wir vollständig auf einen dichtebasierten Clusteralgorithmus. In Konsequenz zur zweidimensionalen Verortung der Datenpunkte entschieden wir uns unter anderem für DBScan als Algorithmus. Durch die wiederholte Anwendung der Clusteranalyse ist unser Verfahren in der Lage, auch Texte mit stark eingestreuten Veränderungen als zusammenhängende Text-Reuse-Instanz zu erkennen.

3.3 Implementierung

Die Implementierung unserer Entwicklung gestaltete sich in einem iterativen Prozess, dessen Meilensteine in diesem Abschnitt übersichtsartig vorgestellt werden.

3.3.1 Prototyping mit Python

Für die prototypische Implementierung des Dotplot-Verfahrens haben wir die Programmiersprache Python eingesetzt. Diese Sprache hat den passenden Funktionsumfang für eine Textanalyse und unterstützt die schnelle prototypische Entwicklung. Die ersten Programmversionen basierten auf einer Zerlegung der Textdokumente in Zeichen-N-Gramme anstatt Wort-N-

Gramme. Unser Verfahren interpretierte dabei jedes Textdokument als einfachen Zeichenstrom, der in einzelne Stücke aufgetrennt wurde. Ein Vergleich der Zeichen-N-Gramme speicherte zunächst nur die erste Position eines identischen Gegenstücks im Vergleichsdokument. Die entstehende Punkteliste wurde nach der x-Koordinate jedes Elements sortiert. Anschließend durchlief eine heuristische Gruppierung diese Liste in festgelegter x-Richtung. Nur direkt benachbarte Punkte wurden in Abhängigkeit eines Schwellwertes zu einer Gruppe zusammengefügt. Die Gruppen wurden danach auf Überschneidungen überprüft und gegebenenfalls zusammengefasst. Daraus resultierten die ersten direkt verwertbaren Cluster.

Die Ergebnisse dieses, an Encoplot angelehnten, Ansatzes blieben jedoch hinter den Erwartungen zurück. Durch die Zeichen-N-Gramme wurden zu viele Übereinstimmungen produziert, von denen aufgrund der Arbeitsweise lediglich die jeweils ersten gemeinsamen Stellen gefunden wurden. Im direkten Vergleich mit einer Dotplot-Visualisierung, fiel die Genauigkeit der automatisch erzeugten Cluster hinter die von Menschen erkannten dichten Bereiche zurück. Zudem wurden durch das Erfassen der lediglich ersten Übereinstimmung eines Zeichen-N-Gramms weitere Informationen vernachlässigt.

Im Zuge des Wechsels zu einer detaillierten Vorverarbeitung der Textdokumente, wurde die Verwendung von Wort-N-Grammen beschlossen. Die Verwendung bereits etablierter Bibliotheken zur Textanalyse und die Vorbereitungen einer Schnittstelle zu einer anderen Bibliothek haben uns dazu bewegt, das bisherige Verfahren in Java umzusetzen. In diesem Zusammenhang konnte ebenfalls die Menge an Werkzeugen zur Textanalyse, dem Information-Retrieval und Data-Mining erweitert werden.

3.3.2 Weiterentwicklung mit Java

Wir setzen während der Textvorverarbeitung auf das *Apache Lucene Framework* [2], dass die initiale Filterarchitektur unserer Verarbeitungs-Pipeline bereitstellt. Als Grundlage der Analysestufe wurden der DBScan-Algorithmus und die Datenstrukturen portiert oder durch Bibliotheken zur Verfügung gestellt. Das gesamte Verfahren baute auf den Möglichkeiten von Java auf und nutzte nur rudimentär externe Bibliotheken. Aufgrund erster Unzulänglichkeiten im Speicherverbrauch und der Laufzeit haben wir relevante Programmmodule identifiziert und ausgelagert. Wichtige Module waren an dieser Stelle zunächst die Datenstruktur für den Clusteralgorithmus DBScan und im weiteren Verlauf DBScan selbst. Es entstand eine Kombination aus Java-basierter Textvorverarbeitung und einer in C++ ausgelagerten Analyse.

3.3.3 Weiterentwicklung mit C++

Die gesamte Analysestufe liegt als natives C++ Programm in einer abgeschlossenen Bibliothek vor. Dafür wurden alle Schritte der Analyse neu imlementiert. Als spezialisierte Datenstruktur für DBScan nutzen wir eine kD-Tree-Implementierung auf Grundlage der Bibliothek *libkdtree++* [23]. Die Anbindung der Analysestufe wird über eine Schnittstelle zur Textvorverarbeitung gewährleistet.

3.3.4 Verteilte Vorverarbeitung mit Hadoop

Unsere Entwicklung wurde anfänglich als monolithische Komponente konzipiert. Dieses Vorgehen hat sich jedoch während der Evaluierung des Verfahrens als hinderlich herausgestellt. Unsere Evaluierung verlangt die Verarbeitung großer Dokumentkorpora. Dadurch steigt die Anzahl der Vergleiche quadratisch in der Anzahl der Dokumente. Zusätzlich wird dieser Faktor noch einmal durch den Vergleich aller Textfragmente innerhalb zweier Dokumente verstärkt. Die Suche gemeinsamer Token innerhalb dieser Datenmenge stellte sich wegen des enormen Speicherverbrauchs und der überdurchschnittlich langen Laufzeit als schwierig heraus. Der ursprünglich monolithisch konzipierte Ansatz musste einem zweigeteilten Verfahren weichen.

Für den paarweisen Dokumentvergleich ändert sich nichts an der beschriebenen Verarbeitung der Dokumente und ihrer Analyse. Die Evaluierung wird jedoch mit der Unterstützung weiterer Technologien durchgeführt. Der bereits angesprochene, intensive Dokumentvergleich kann durch eine Verteilung auf mehrere Maschinen beschleunigt werden. Weiterhin ist es durch eine persistente Speicherung der Zwischenergebnisse möglich, dieses Vorwissen wiederzuverwenden. Die Zwischenergebnisse setzen sich insbesondere aus den Token und Positionen zusammen. Dafür verwenden wir das Map-Reduce-Framework *Apache Hadoop* [1].

Die Verarbeitung der Token erfolgt durch Map-Reduce-Jobs. Alle Token der Korpusdokumente werden mit ihren Positionsinformationen in einer Liste zusammengefasst. Diese Liste wird anschließend so weit verdichtet, dass lediglich mehrmals vorkommende Token in der Liste verbleiben. Jeweils gleiche Listeneinträge werden kombiniert und mit den Positionsinformationen, die inzwischen Punkten entsprechen, herausgeschrieben. Das Framework sorgt dafür, die notwendigen Daten auf die parallelisierten Map-Reduce-Jobs zu verteilen. In den Ergebnissen dieses Prozesses sind alle gemeinsamen Token innerhalb der Dokumente eines Korpus inklusive ihrer Positionen verzeichnet und in wiederverwendbarer Form gespeichert.

3.3.5 Komponente einer Bibliothek

Unser Verfahren wurde während seiner Entwicklung sowohl als separates Programm als auch als Komponente eines größeren Systems verstanden. Aus diesem Grund existieren definierte Schnittstellen zu einer Bibliothek, die weitere Einzelkomponenten aus dem Information-Retrieval bereitstellt. Die *AITools*-Bibliothek [43] umfasst breit gefächerte Funktionalitäten aus dem Information-Retrieval und Data-Mining. Mit Unterstützung dieser einzelnen Werkzeuge können neue IR-Systeme aufgebaut werden. Unsere Entwicklung versteht sich als eine Komponente dieser Bibliothek.

Kapitel 4

Evaluierung

In diesem Kapitel stellen wir die von uns durchgeführten Experimente und Beobachtungen vor. In diesem Zusammenhang werden die notwendigen Gütemaße, die Parameter unseres Verfahrens und die verwendete Datengrundlage erläutert. Daraufhin folgt eine Beschreibung der durchgeführten Experimente bezüglich änderbarer Parameter. Die Ergebnisse der Experimente und die daraus resultierende Parameterentwicklung folgen im Anschluss. Die Bewertung der einzelnen Parameter ist essentiell für die Benutzbarkeit unseres Verfahrens. Deshalb liegt das Hauptaugenmerk auf einer Evaluierung gebräuchlicher Parametereinstellungen, die zu verwertbaren Ergebnissen innerhalb einer Text-Reuse-Analyse führen.

4.1 Referenzdaten

Das Aufgabenspektrum einer Text-Reuse-Analyse der einer Plagiatanalyse. Die Plagiaterkennung ist in diesem Sinne als Untermenge der Text-Reuse-Erkennung zu verstehen. Anhand der Charakterisierung von Text-Reuse und der damit verbundenen Analyse kann eine Verfahrensähnlichkeit zur Plagiaterkennung festgestellt werden. Unter der Prämisse, auch bei der Evaluierung auf erprobte Mittel zurückzugreifen, wurde keine separate Datenbasis aufgebaut. Wir haben uns an der Ähnlichkeit zur Plagiatanalyse orientiert und auf einem generischen Framework zur Evaluierung von Plagiatdetektoren aufgesetzt.

Als erstes Exemplar seiner Art, für groß angelegte Experimente und Evaluierungen von Plagiatdetektoren, hat sich das *PAN-Plagiatskorpus (PAN-PC)* als geeignet hervorgetan [8, 29]. Das Korpus ist eine zusammengestellte Menge unterschiedlich stark veränderter Textdokumente, die verschiedene Varianten von Plagiaten und anderer Textwiederverwendung vereinen. Gleichzeitig werden die Dokumente durch ausgeprägte Metainformationen mit zusätzlichen Daten angereichert, die unerlässlich für eine Evaluierung sind.

Bestehend aus 27073 Dokumenten, teilt sich das Korpus zu je 50% in Quelldokumente (*source documents*) und Verdachtsdokumente (*suspicious documents*). Letztere sind zu gleichen Teilen untergliedert in Dokumente mit und ohne Textwiederverwendung.

Das PAN-Framework führt verschiedene Möglichkeiten eines Plagiats in seinem Konzept ein. Diese sollen die Plagiierung als Anwendung von Texttransformationen auf einen Originaltext, zum Zwecke der nicht legitimierten Ausgabe als eigenes Produkt, modellieren helfen. An dieser Stelle ist die einfache Kopie eines Textes, die naivste Variante einer Transformation. Weitere Möglichkeiten haben wir in diesem Zusammenhang bereits besprochen.

Im Framework wird die Glaubwürdigkeit eines Plagiats, im Sinne der Evaluierung, in die drei Varianten echtes, simuliertes und künstliches Plagiat untergliedert. Durch eine unklare rechtliche und ethische Lage in Bezug auf echte Plagiate wurde es notwendig, einen adäquat verwendbaren Ersatz zu bestimmten. Simulierte Plagiate sind von Menschenhand erstellte Texte, die aus einer Paraphrasierung vorgegebener Originaltexte entstanden sind. Sowohl echte als auch simulierte Plagiate unterscheiden sich lediglich in der Intention der Autoren. Für die rein technische Evaluierung der Algorithmen mit diesen Texten können beide Varianten als gleichwertig betrachtet werden. Als weitere Variante werden künstliche Plagiate innerhalb der Textdokumente verwendet. Diese Plagiate entstehen auf automatischem Weg durch eine Software. Durch die Anwendung unterschiedlicher Texttransformationen produziert die Software künstlich obfuskierte Derivate der Originaltexte. Bei diesem Vorgehen wird versucht, die semantische Verbindung zum Original mithilfe der Analyse von Text- und Wortmerkmalen zu erhalten [8, 29].

Alle Dokumente des PAN-PC sind mit Metainformationen ausgezeichnet, die grundsätzlich Aufschluss über Herkunft und Sprache geben. Verdächtige Dokumente mit tatsächlichen Text-Reuse-Instanzen verzeichnen in ihren zugehörigen Metainformationen eine genaue Auflistung dieser Plagiatstellen.

Erzeugung eines Testkorpus

Unser iterativer Entwicklungsansatz erforderte die schnelle Überprüfung der einzelnen Verfahrenskomponenten und des Algorithmus im Ganzen. Wir haben deshalb einen kleinen Testkorpus aus der Dokumentmenge des PAN-PC extrahiert. Dadurch erstreckten sich die Experimente über einen Bruchteil der Zeit im Vergleich zur Analyse auf dem PAN-PC.

Zunächst fokussierten wir die Entwicklung unseres Verfahrens auf die Analyse englischsprachiger Textdokumente. Damit bekam der Testkorpus Beschränkungen auferlegt, denen die enthaltenen Dokumente entsprechen mussten. 1000 verdächtige Dokumente mit 9875 explizit ausgezeichneten Plagiat-

stellen wurden aus dem PAN-PC-10 zufällig gezogen. Die zugehörigen Metainformationen der Dokumente mussten der Bedingung genügen, Englisch als einzige Dokumentsprache aufzuweisen. Dies galt ebenfalls für die referenzierten Quelldokumente. Anschließend wurden 1852 referenzierte Quelldokumente aus den Metainformationen der Plagiatstellen extrahiert. Für eine gleichmäßige Durchmischung der Korpusdokumente wurden daraufhin ebenso viele zufällige Quelldokumente gezogen. Gleiches galt für die verdächtigen Dokumente. Weitere 1000 zufällige Dokumente ohne explizite Plagiatauszeichnungen sind Bestandteil des Korpus. Damit ergibt sich eine absolute Zahl von 2000 verdächtigen Dokumenten und 3704 Quelldokumenten.

4.2 Gütemaße

Das PAN-Framework definiert für die Bewertung der Güte eines Plagiatdetektors die drei grundlegenden Maße *Precision*, *Recall* und *Granularity*. Diese Gütemaße setzen die erkannten Textstellen eines Plagiatdetektors in Relation zu den vorgegebenen Plagiatstellen des PAN-PC. Jede als wiederverwendet bzw. plagiiert erkannte Textstelle und jede Referenzstelle werden innerhalb des PAN-Frameworks als Menge von Zeichen repräsentiert. Das Dokument d_{plg} sei ein Dokument mit plagiierten Textstellen. Eine Plagiatstelle s in d_{plg} wird als 4-Tupel wie in (4.1) definiert.

$$s = \langle s_{plg}, d_{plg}, s_{src}, d_{src} \rangle \tag{4.1}$$

$$r = \langle r_{plg}, d_{plg}, r_{src}, d'_{src} \rangle \tag{4.2}$$

Mit s_{plg} und s_{src} werden die plagiierte Textstelle aus dem verdächtigen Dokument d_{plg} und die Originalstelle aus dem Quelldokument d_{src} repräsentiert [29]. Die Plagiatstellen überschneiden sich diesbezüglich nicht. Die Erkennung r einer Plagiatstelle s im Verdachtsdokument d_{plg} ist ebenfalls ein 4-Tupel nach (4.2) mit gleichem Aufbau. Mit r_{plg} als erkannter plagierter Textstelle und r_{src} als erkannter Referenz im vermutlichen Quelldokument d'_{src}.
Eine Erkennung wird durch das PAN-Framework als gültig definiert, wenn die Bedingungen in (4.3) zutreffen [29].

$$\begin{aligned} d'_{src} &= d_{src} \\ r_{plg} \cap s_{plg} &\neq \emptyset \\ r_{src} \cap s_{src} &\neq \emptyset \end{aligned} \tag{4.3}$$

Die Menge S umfasst die in den Metainformationen ausgezeichneten Plagiatstellen $s \in S$, währenddessen in R alle durch ein Analyseverfahren produzierten Erkennungen $r \in R$ vereint sind.

Die Erkennung wiederverwendeter Textstellen ist stark von der Größe der einzelnen Abschnitte abhängig. Ihre Interpretation als zusammengehörige Text-Reuse-Instanz kann entweder durch den Menschen oder aber direkt durch den Algorithmus erfolgen. Somit können die Erkennungen eines Algorithmus sowohl einzelne als auch mehrere wiederverwendete Textstellen unfassen. Unsere Entwicklung fasst durch ein globales Sequence-Alignment nah verortete Textstellen zu einer Text-Reuse-Instanze zusammen. Es ist bestrebt, so wenige fragmentierte Text-Reuse-Instanzen wie möglich zu erzeugen. Das PAN-Framework stellt für diese Herangehensweise der Textanalyse bereits angepasste Gütemaße bereit. Precision, Recall und Granularity werden in Bezug zur absoluten Zeichengröße der erkannten Stellen berechnet. Über die Repräsentation eines Dokuments als Menge $\mathbf{d} = \{(1, d), \cdots, (|d|, d)\}$ kann jedes Zeichen i in diesem Dokument separat durch ein Tupel (i, d) referenziert werden. Eine Plagiatstelle s repräsentiert damit eine Vereinigung der zum Plagiat gehörenden Zeichen des Originals und des plagiierten Textes (4.4). Dies gilt gleichfalls für eine Erkennung r nach den Bedingungen in (4.5).

$$\begin{aligned} \mathbf{s} &= \mathbf{s}_{plg} \cup \mathbf{s}_{src} \qquad (4.4) \\ & \quad \mathbf{s}_{plg} \subseteq \mathbf{d}_{plg} \ \wedge \ \mathbf{s}_{src} \subseteq \mathbf{d}_{src} \\ \mathbf{r} &= \mathbf{r}_{plg} \cup \mathbf{r}_{src} \qquad (4.5) \\ & \quad \mathbf{r}_{plg} \subseteq \mathbf{d}_{plg} \ \wedge \ \mathbf{r}_{src} \subseteq \mathbf{d}'_{src} \end{aligned}$$

Die Vollständigkeit der Erkennungen wird als Gütemaß durch den *Recall* ausgedrückt. Dieses Maß setzt die produzierten gültigen Erkennungen in Relation zu den ausgezeichneten Plagiatstellen der Korpusinformationen (4.6). Die Genauigkeit der Ergebnisse wird durch die *Precision* erfasst. Precision setzt die gültigen Erkennungen eines Verfahrens in Beziehung zu allen Erkennungen des Verfahrens (4.7).

$$recall(S, R) = \frac{\left| \bigcup_{(s,r)\in(S\times R)} (\mathbf{s} \sqcap \mathbf{r}) \right|}{\left| \bigcup_{s\in S} \mathbf{s} \right|} \qquad (4.6)$$

$$precision(S, R) = \frac{\left| \bigcup_{(s,r)\in(S\times R)} (\mathbf{s} \sqcap \mathbf{r}) \right|}{\left| \bigcup_{r\in R} \mathbf{r} \right|} \qquad (4.7)$$

$$\mathbf{s} \sqcap \mathbf{r} = \begin{cases} \mathbf{s} \cap \mathbf{r} & \text{Wenn } r \text{ eine gültige Erkennung von } s \text{ ist.} \\ \emptyset & \text{Sonst.} \end{cases}$$

Ein weiteres Konzept wird durch das PAN-Framework in Bezug auf die Granularität der erkannten Plagiatstellen eingeführt. Die Leistungsfähigkeit eines Verfahrens ist ebenfalls davon abhängig, ob Plagiatstellen zusammenhängend oder stückweise erkannt werden [29]. Aus diesem Grund wird vom PAN-Framework eine Bewertung für diese Eigenschaft der Ausgabedaten definiert. An eine gültige Erkennung wird die Bedingung gestellt, eine ausgezeichnete Plagiatstelle möglichst vollständig zu erkennen. *Granularity* (4.8) macht diese Bedingung messbar.

$$granularity(S, R) = \frac{1}{|S_R|} \sum_{s \in S_R} |R_s| \tag{4.8}$$

$$\begin{aligned} S_R &= \{s | s \in S \wedge \exists r \in R : r \text{ erkennt } s\} \\ R_s &= \{r | r \in R \wedge r \text{ erkennt } s\} \end{aligned}$$

Dabei bezeichnet $S_R \subseteq S$ die Plagiatstellen, für die gültige Erkennungen in R existieren. Alle produzierten gültigen Erkennungen einer bestimmten Plagiatstelle s werden durch R_s repräsentiert. Dieses Gütemaß bewegt sich im Wertebereich $[1, |R|]$, wobei die untere Grenze eine optimal zusammenhängende Abdeckung der Plagiatstellen S durch die produzierten Erkennungen repräsentiert, steigende Werte hingegen eine verstärkte Fragmentierung widerspiegeln.

Die drei Einzelmaße Precision, Recall und Granularity werden durch das PAN-Framework zu dem Effizienzwert *Plagdet* vereinigt (4.9). Dieser setzt das harmonische Mittel (F_1-*Measure*) von Precision und Recall ins Verhältnis zur Granularität der produzierten Erkennungen.

$$plagdet(S, R) = \frac{F_1}{log_2(1 + granularity(S, R))} \tag{4.9}$$

Mit diesem Wertungssystem können unterschiedliche Plagiatdetektoren und in unserem Fall auch ein Verfahren zur Text-Reuse-Analyse gegeneinander verglichen werden. In vorangegangenen Arbeiten [42] zeigte sich bereits, dass das Interesse an der Entwicklung leistungsfähiger Plagiatdetektoren zu einer wachsenden Vergleichsmenge an Verfahren führt. Diesbezüglich können weitere Beobachtungen in unterschiedlichen Experimenten angestellt werden.

4.3 Parameterzusammenstellung

Unser Textanalysesystem besteht aus zwei Verarbeitungsstufen. Die Textvorverarbeitung sorgt durch eine Normalisierung für einheitliche Eingabedaten. Die Datenanalyse verwertet diese Informationen und berechnet Marken zur

Textextraktion. Zur Kontrolle des Verhaltens während der Datenverarbeitung existieren zehn Parametern. Diese werden im Folgenden vorgestellt und im anschließenden Abschnitt zu den Experimenten in ihrer Ausprägung und Einflussnahme auf die Erkennungsleistung des Verfahrens untersucht.

4.3.1 Parameter der Textvorverarbeitung

Fast jeder der beschriebenen Schritte zur Textnormalisierung bietet die Möglichkeit einer Wertveränderung. Die Normalisierung wurde der Filteranwendung ähnelnd stückweise aufgebaut. Als fester Teil und Grundlage für die anschließende Textverarbeitung ist die Tokenisierung nicht veränderbar. Ebenfalls der Chunking-Filter zur Erzeugung der Wort-N-Gramme (*Chunks*) ist in seiner Anwendung ein Pflichtteil der Filterkette. Jedoch bietet er die Möglichkeit, die Anzahl der Wörter N jedes extrahierten Textfragments einzustellen. Die weiteren Einstellungen gliedern sich in binäre Schalter oder direkte Wertangaben und sind schrittweise zuschaltbar.

Über die Notwendigkeit der Entfernung von Stoppworten wurde bereits in vergangenen Abschnitten gesprochen. Um dennoch die Auswirkungen der Stoppwortentfernung festzuhalten wurde dieser Filter als optionale Normalisierung eingerichtet.

Darüber hinaus ist der Stemming-Filter in seiner Einstellung ebenfalls optional zuschaltbar. Dieser Filter untergliedert sich in die insgesamt drei Optionen zum Einsatz eines Snowball-Stemmers, dem Suffix-Stemming oder keinem von beiden. Unter dem vorausgesetzten Einsatz eines einfachen Suffix-Stemming, kann hier zusätzlich die Anzahl der zu erhaltenden Zeichen pro Wortanfang eingestellt werden.

Ein weiteres noch ungenutztes Merkmal der Textnormalisierung ist die Ersetzung einzelner Wörter durch die zugehörigen Synonyme. Der optionale Synonym-Filter wird gleichmäßig auf alle Wörter angewendet und ersetzt jedes Wort abhängig von den Einstellungen durch dessen Synonym. Aufgrund der Priorisierung auf eine schnelle Vorverarbeitung wurde zunächst auf diesen Filter verzichtet. Das Nachschlagen der Synonyme in der Datenbank lag zum Zeitpunkt dieser Entwicklung weit über dem geforderten Zeitlimit. Mit dem Einsatz der Synonymdatenbank WordNet muss in zukünftigen Programmversionen die Anbindung dieser externen Datenquelle geprüft werden. Für diese Evaluierung wurde der Parameter nicht weiter betrachtet.

Als letzte verwendbare Einstellung für die Textnormalisierung lässt sich die Sortierung der Wörter jedes Chunks ebenfalls optional zuschalten.

Damit besteht die Menge veränderbarer Parameter aus fünf einzelnen Einstellungen. Die Wertebereiche sind differenziert zu den in den Experimenten erreichten Effizienzwerten zu bewerten und werden später erneut aufgegriffen.

4.3.2 Parameter der Datenanalyse

Die Datenanalyse besitzt als Recheneinheit des Verfahrens die fünf Parameter zur Einstellung der beiden Clusteranalyseschritte und der Clusterfilterung. Die Filterung der Cluster aus der ersten Clusteranalyse nach ihrer absoluten Größe erfolgt durch einen einfachen Schwellwert. Seine Einstellung bezieht sich direkt auf die Anzahl der durch das Cluster eingeschlossenen Zeichen. So führt der Schwellwert zu einer Säuberung zu kleiner Cluster aus der Datenmenge und verringert dadurch das andernfalls vermehrt auftretende Rauschen.

Grundsätzlich existieren für die beiden DBScan-Instanzen die notwendigen Einstellungen für den Radius der ϵ-Nachbarschaft und der Mindestzahl benachbarter Punkte $MinPts$ eines Kernpunktes. Der verwendete Clusteralgorithmus DBScan ist aufgrund seiner Architektur kein selbstadaptives Verfahren in Bezug auf die Eingangsdaten. Diese Notwendigkeit erfordert vorerst eine manuelle Evaluierung der Wertebereiche. Ohne eine spezifische Bevorzugung bestimmter Wertebereiche bezüglich der zur Evaluierung verwendeten Daten muss ein Mittelweg zwischen akzeptabler Güte des Verfahrens und geforderter Beliebigkeit der Eingangsdaten gewährleistet werden.

4.4 Experimente

Als Grundlage unserer Evaluierung dient das bereits vorgestellte Testkorpus. Alle Experimente werden nach einem festgelegten Prozess durchgeführt und sind prinzipiell unabhängig vom verwendeten Korpus. Die einzige Bedingung ist der übereinstimmende Strukturaufbau zum PAN-PC, insbesondere der Metainformationen der einzelnen Dokumente.

4.4.1 Experimentbeschreibung

Jedes Experiment besteht aus einem vollständigen Vergleich aller verdächtigen Dokumente mit allen Quelldokumenten eines Korpus und der Untersuchung auf gemeinsame Textabschnitte. Der Evaluierungsprozess für diese Aufgabenstellung setzt sich aus vier Schritten zusammen.

Textnormalisierung und Datenrepräsentation

Durch das Apache Lucene Framework erfolgt die Normalisierung der Textdokumente. Die Dokumentrepräsentation wird anschließend zur Wiederverwendung persistent gespeichert.

Berechnung gemeinsamer Token mit Hadoop

Die Dokumentrepräsentationen aller Korpusdokumente werden in das Hadoop Dateisystem geladen. Jedes Token ist eindeutig in der Menge aller Verdachts- und Quelldokumente referenziert. Eine Token-ID ist ein 4-Tupel $\langle d, l, \mathbf{p}, t\rangle$, das sich aus Herkunftsdokument (d), Länge (l) der nicht normalisierten Textstelle, Positionsliste ($\mathbf{p}$) aller Erscheinungen dieser Textstelle in d und dem Token t zusammensetzt.

Drei Map-Reduce-Instanzen verarbeiten diese Datenmenge und extrahieren daraus die gemeinsamen Token aller verdächtigen Dokumente und Quelldokumente. Zu Beginn wird eine Liste aus Token-IDs erstellt und alle identischen Token anhand der beiden Dokumenttypen d_{plg} und d_{src} aufgeteilt. Anschließend werden die Tripel beider Dokumenttypen für alle identischen Token kombiniert und in Tupeln $\langle\langle d_{plg}, l_{plg}, \mathbf{p}_{plg}\rangle, \langle d_{src}, l_{src}, \mathbf{p}_{src}\rangle\rangle$ abgelegt. Ein kartesisches Produkt der Positionslisten $\mathbf{p}_{src}$ und $\mathbf{p}_{plg}$ aller vorher erzeugten Tupel berechnet die gewünschte vollständige Kombination aller gemeinsamen Token.

Die resultierende Beziehung zwischen beiden Dokumenttypen ist eine Menge von 6-Tupeln $\langle d_{plg}, l_{plg}, p_x, d_{src}, l_{src}, p_y\rangle$ für jede Tokenposition (p_x, p_y) mit $p_x \in \mathbf{p}_{plg}$ und $p_y \in \mathbf{p}_{src}$. Jedes 6-Tupel repräsentiert ein identisches Token bezüglich zweier Dokumente d_{plg} und d_{src}. Als letzte Phase der Hadoop-Verarbeitung werden diese Abbildungen persistent gespeichert und der Analysestufe übergeben.

Analyse der Dichteverteilung gemeinsamer Token

Die Tokenrepräsentationen werden nach der Signatur der verdächtigen Dokumente verarbeitet. Mit der Verfügbarkeit der absoluten Position bezüglich zweier Dokumente kann die Analyse das vom Dotplot abgeleitete Punktefeld aufbauen. Dieses wird entsprechend der beschriebenen Schritte analysiert und die Ergebnisse gespeichert. Für jedes Dokumentpaar aus verdächtigem Dokument und Quelldokument wird somit eine Clustermenge als Vergleichsresultat berechnet.

Leistungsbewertung durch Güteberechnung nach PAN-Framework

Die letzte Stufe jedes Experiments vergleicht die Daten aus den Metainformationen des Korpus mit den Resultatswerten der vorangegangenen Analyse. Die vorgestellten Gütemaße werden auf den extrahierten Daten berechnet und erlauben eine Bewertung der Leistungsfähigkeit des Verfahrens.

Stoppwortentfernung	Inaktiv
Stemming-Typ	Suffix-Stemming
Stemming-Option	5 Zeichen
Synonymersetzung	Inaktiv
Wort-N-Gramm-Sortierung	Inaktiv
DBScan 1 ϵ-Radius	1000
DBScan 1 $MinPts$	4
Clusterfilter	90 Zeichen
DBScan 2 ϵ-Radius	3000
DBScan 2 $MinPts$	4

Tabelle 4.1. Initiale Parameterausprägungen für den Beginn der Experimente.

4.4.2 Experimente auf dem Testkorpus

Das beschriebene Testkorpus wurde über die prototypische Entwicklung hinaus, als Evaluierungsgrundlage für unser Verfahren verwendet. Eine schnelle Analyse war essenziell, um die Menge an möglichen Parameterausprägungen im Ansatz zu untersuchen und den Wertebereich einzuschränken. Wir haben uns daher entschlossen, unser Verfahren schrittweise mit dieser Dokumentmenge durch stetige Experimente weiterzuentwickeln. Folglich basieren die erarbeiteten Parameterausprägungen auf einer Trendanalyse. Diese wurde ausgehend von der Effizienz des Verfahrens auf dem Testkorpus durchgeführt.

Als Startpunkt der Evaluierung wurden durch sinnvolle Annahmen Vorgabewerte ausgewählt, um mit den ersten Experimenten zu beginnen. Die Effizienz jedes Experimentes war Grundlage der Bewertungen für die darauf folgenden Experimente. In Tabelle 4.1 sind die ersten Parametereinstellungen zusammengetragen. Die Stoppworte wurden zunächst nicht entfernt, da diese Einstellung Augenmerk des ersten Experiments war. Das Suffix-Stemming wurde auf eine feste Präfixlänge von fünf Zeichen gesetzt und sollte die mittlere Wortlänge der englischen Sprache repräsentieren [33, 34]. Das Chunking war zunächst auf Wort-5-Gramme eingestellt. Eine Sortierung fand diesbezüglich nicht statt. Die beiden Instanzen zur Clusteranalyse bekamen zunächst moderate Radiuswerte. Beide DBScan-Instanzen wurden in den Einstellung mit Erwartungswerten für lokales und globales Alignment vorbelegt. Unsere Erwartung hinter dieser ersten Vorauswahl war eine möglichst zusammenhängende Erkennung wiederverwendeter Textabschnitte. Die Filterung der Cluster zwischen den beiden Clusteranalysen sollte zunächst Textabschnitte von ungefähr zwei Satzlängen behalten. Alle darunter liegenden kleineren Textabschnitte erschienen uns als zusätzliches Rauschen und sollten von Beginn an unterdrückt werden.

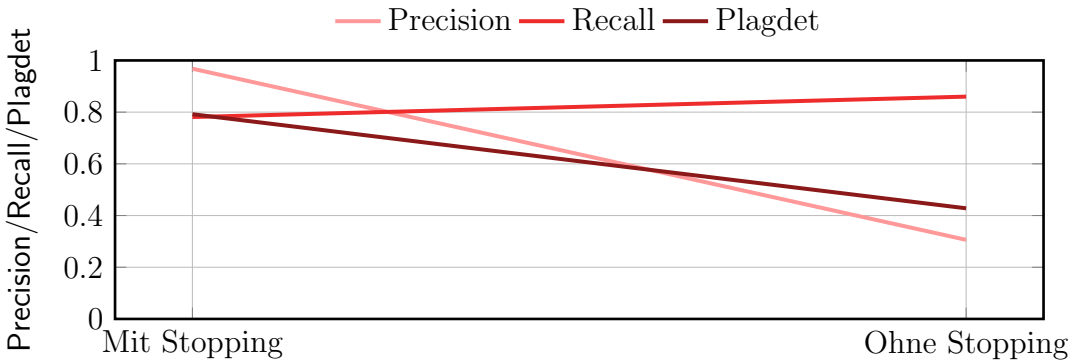


Abbildung 4.1. Die Einwirkung der Stoppwortentfernung (*Stopping*) auf die Effizienz drückt sich deutlich durch den Fall aller Leistungswerte bei abgeschalteter Entfernung aus.

In den nachfolgenden Abschnitten werden die Experimente anhand der Stichprobenergebnisse der Leistungsbewertung aufgelistet. Die Synonymersetzung ist kein Bestandteil der Experimente.

Stoppwortentfernung

Als erstes Experiment wurde die Stoppwortentfernung (*Stopping*) getestet. Die Ergebnisse in Tabelle 4.2 spiegeln jedoch bereits den unsererseits erwarteten Effekt wieder, dass die Entfernung der Stoppworte die Genauigkeit der Ergebnisse positiv beeinflusst. Die Unterschiede in der Anwendung des Verfahrens sind in Abbildung 4.1 visualisiert.

Durch den Erhalt der Stoppworte steigt die Menge der gefundenen Textabschnitte. Jedoch fällt durch die zusätzlich übereinstimmenden Abschnitte und aufgrund der Menge irrelevanter Passagen ohne ausgezeichnete Korpusinformationen die Genauigkeit. Trotz der Vielzahl zusätzlicher Textabschnitte werden die relevanten Teile ohne große Fragmentierung der Ergebnisse gefunden.

Zunächst legten wir die Priorität darauf, dass Recall und Precision in gleichem Maße auf einen Wert von mehr als 0,5 steigen. Dadurch sollte das Ver-

	Precision	Recall	Gran.	Plagdet	Cases	Det.
Ohne Stopping	0,306	0,859	1,075	0,428	9875	29454
Mit Stopping	0,968	0,780	1,131	0,792	9875	9211

Tabelle 4.2. Die Ergebnisse des Experiments zeigen deutliche Unterschiede in der Genauigkeit. Die Anzahl der Erkennungen (Det.) spiegelt den vergleichsweise guten Recall wieder. Beide Ergebnisse weisen eine gute Granularität (Gran.) nahe bei Eins auf.

fahren mit größerer Sicherheit analysiert werden können. Dieses Ziel erreichten wir, indem fortan die Stoppworte dauerhaft entfernt wurden.

Stemming-Varianten

Grundlage dieses Experiments war die Beobachtung der Leistungswerte anhand wechselnder Stemming-Varianten. In diesem Zusammenhang haben wir den regelbasierten Porter-Stemmer und ein einfaches Suffix-Stemming eingesetzt. Als dritte Auswahlmöglichkeit blieb der Verzicht auf jegliches Stemming. Die Ergebnisse in Tabelle 4.3 zeigen nur marginale Unterschiede bei den Leistungswerten.

	Precision	Recall	Gran.	Plagdet	Cases	Det.
Kein Stemming	0,974	0,778	1,131	0,792	9875	9132
Porter-Stemming	0,969	0,780	1,131	0,792	9875	9203
Suffix-Stemming	0,969	0,781	1,131	0,792	9875	9211

Tabelle 4.3. Effizienz drei verschiedener Stemming-Varianten.

Auf den Porter-Stemmer wurde verzichtet, weil seine Sprachabhängigkeit keinen bevorzugenden Vorteil gegenüber dem einfachen Suffix-Stemming oder gar keinem Stemming hatte. Vielmehr war die Verarbeitungsgeschwindigkeit von deutlichem Unterschied im Vergleich zu den anderen beiden Varianten.

Der vollständige Verzicht auf Stemming erbrachte gute Ergebnisse auf dem Testkorpus. Im Vergleich dazu konnte das Suffix-Stemming ebenfalls mit gutem Recall und starker Precision aufwarten.

Die endgültige Entscheidung nach diesem Experiment fiel auf das Suffix-Stemming. Ausschlaggebend dafür war zunächst die Möglichkeit zur Einsparung des Datenvolumens während der Analyse, bei gleichbleibenden und teilweise besseren Ergebnissen der Gütemaße. Weiterhin ermöglichte das Suffix-Stemming auf einfache Weise eine ähnliche Rückführung auf die Stammform eines Wortes wie der Porter-Stemmer, allerdings mit erheblich flexiblerer Steuerung durch einen Schwellwert.

Suffix-Stemming

Das Suffix-Stemming wird über einen diskreten positiven Schwellwert gesteuert. In diesem Experiment (Tabelle 4.4) wurde die Länge der verbleibenden Wortpräfixe variiert. Unter der starken Verkürzung der einzelnen Wörter tauchen mit verringerter Präfixlänge ähnliche Token häufiger in der Vergleichsmenge auf. Die hier durchgeführen Experimente bezogen sich auf eine fixe Einstellung des Chunkings. Das verwendete Wort-5-Gramm verursachte in Verbindung mit dem variierten Stemming-Parameter

WGram	Stem	Precision	Recall	Gran.	Plagdet	Cases	Det.
5	2	0,945	0,786	1,129	0,787	9875	9489
	3	0,965	0,783	1,131	0,792	9875	9279
	4	0,967	0,782	1,131	0,792	9875	9245
	5	0,969	0,781	1,131	0,792	9875	9211
4	2	0.766	0.813	1.112	0.732	9875	11811
	3	0.869	0.810	1.114	0.776	9875	10416
	4	0.889	0.809	1.115	0.784	9875	10171
	5	0.893	0.807	1.115	0.784	9875	10109

Tabelle 4.4. Die Präfixlängen des Suffix-Stemming und ihre zugehörigen Leistungswerte auf dem Testkorpus. *Stem* bezeichnet die Anzahl der verbliebenen Zeichen pro Wort.

keine große Veränderung in den Leistungswerten des Algorithmus. Im Vergleich dazu haben die Präfixlängen anders in Kombination mit einem Wort-4-Gramm auf die Anzahl der Erkennungen gewirkt und damit auch die Qualität der Gütemaße beeinflusst (Abb. 4.2).

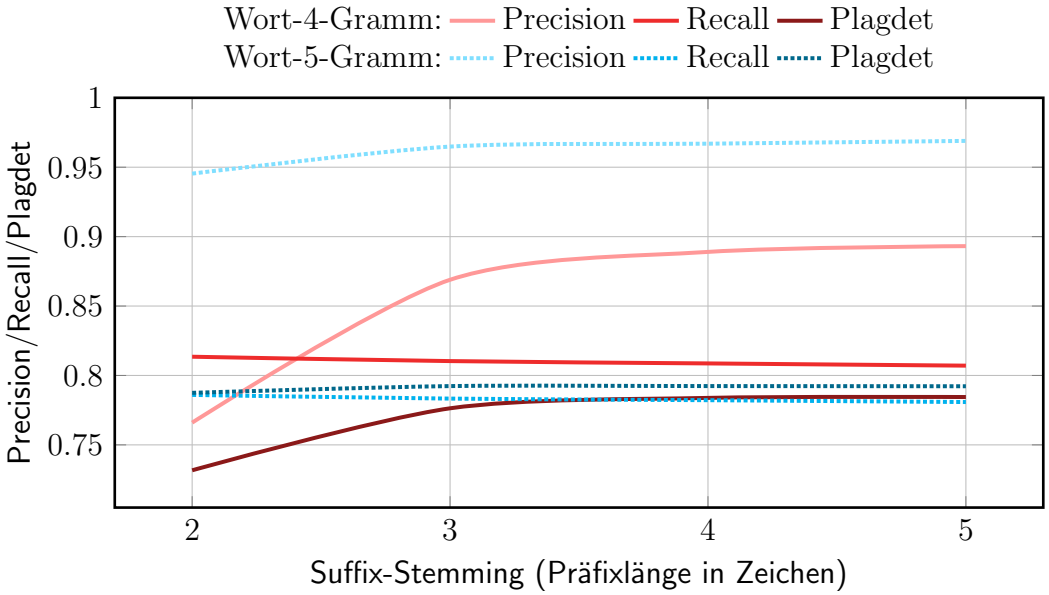


Abbildung 4.2. Die Variation der Präfixlängen beim Suffix-Stemming in Verbindung mit unterschiedlichen Chunklängen zeigt eine steigende Genauigkeit ab drei Anfangszeichen für jedes Wort.

Wir haben uns nach diesem Experiment dafür entschlossen, die zukünftigen Untersuchungen stets mit einer zusätzlichen Variation der Präfixlänge zu versehen. Das erlaubte eine ausführlichere Analyse in Kombination mit weiteren

Parametern der Vorverarbeitung. Die beiden Präfixlängen 4 und 5 wiesen ein vielversprechendes Verhältnis zwischen Recall und Precision auf. Im Vergleich dazu sind kleinere Präfixlängen ebenfalls gut für die Gesamtleistung und verringern das Datenaufkommen. Wir erwarten jedoch, dass die Zahl der dadurch übereinstimmenden Token auf großen Korpora wie dem PAN-PC erheblich zunimmt und dadurch die Genauigkeit schnell sinkt.

Chunking

Mit den ausgewählten Präfixlängen wurde der Wertebereich für das Chunking evaluiert, dessen Experimentergebnisse in Tabelle 4.5 nachzulesen sind.

Stem	WGram	Precision	Recall	Gran.	Plagdet	Cases	Det.
3	4	0,869	0,810	1.114	0,776	9875	10416
	5	0,965	0,783	1.131	0,792	9875	9279
	6	0,981	0,761	1.156	0,773	9875	9143
	10	0,995	0,661	1.364	0,640	9875	9857
5	4	0,893	0,807	1,115	0,784	9875	10109
	5	0,969	0,780	1.131	0,792	9875	9211
	6	0,984	0,759	1.157	0,773	9875	9112
	10	0,996	0,660	1.365	0,639	9875	9836

Tabelle 4.5. Der Testkorpus wurde mit unterschiedlichen Chunklängen verarbeitet und in Korrelation mit den Stemming-Einstellungen analysiert. *WGram* bezeichnet in diesem Fall direkt die Wort-N-Gramme und ihre entsprechende Länge in Wörtern.

Die Veränderung der Gesamtleistung auf dem Testkorpus ist nur unmerklich verschieden zwischen den benachbarten Parametereinstellungen. Jedoch sinkt der Recall um knapp 2% bei der Verwendung eines Wort-6-Gramms mit einer Präfixlänge von 4 Zeichen pro Wort. Die Länge des Tokens wirkt sich hier nachteilig auf den Recall aus, da dieser Parameter eine Spezialisierung der Token im gegenseitigen Vergleich bewirkt. Die Wahrscheinlichkeit ähnlich lange Token zu finden sinkt damit rapide. Gleichzeitig wird jedoch die Genauigkeit erhöht und die Sicherheit des Verfahrens in Bezug auf relevante Textbestandteile verbessert. Erhärtet wird diese Vermutung durch das in der Tabelle nachzulesende Beispiel eines Wort-10-Gramms,das als Negativbeispiel die Wirkung verdeutlichen sollte.

Es hat sich ebenfalls herausgestellt, dass die Präfixlänge und die Anzahl der Wörter innerhalb jedes Tokens miteinander korrelieren. Kurze Wortpräfixe wirken der Spezialisierung langer Chunks entgegen und können eine größere Anzahl ähnlicher Token vergleichbar machen. Im umgekehrten Fall können kurze Chunks durch längere Wortpräfixe besser für Überprüfungen auf exakte Übereinstimmungen eingesetzt werden.

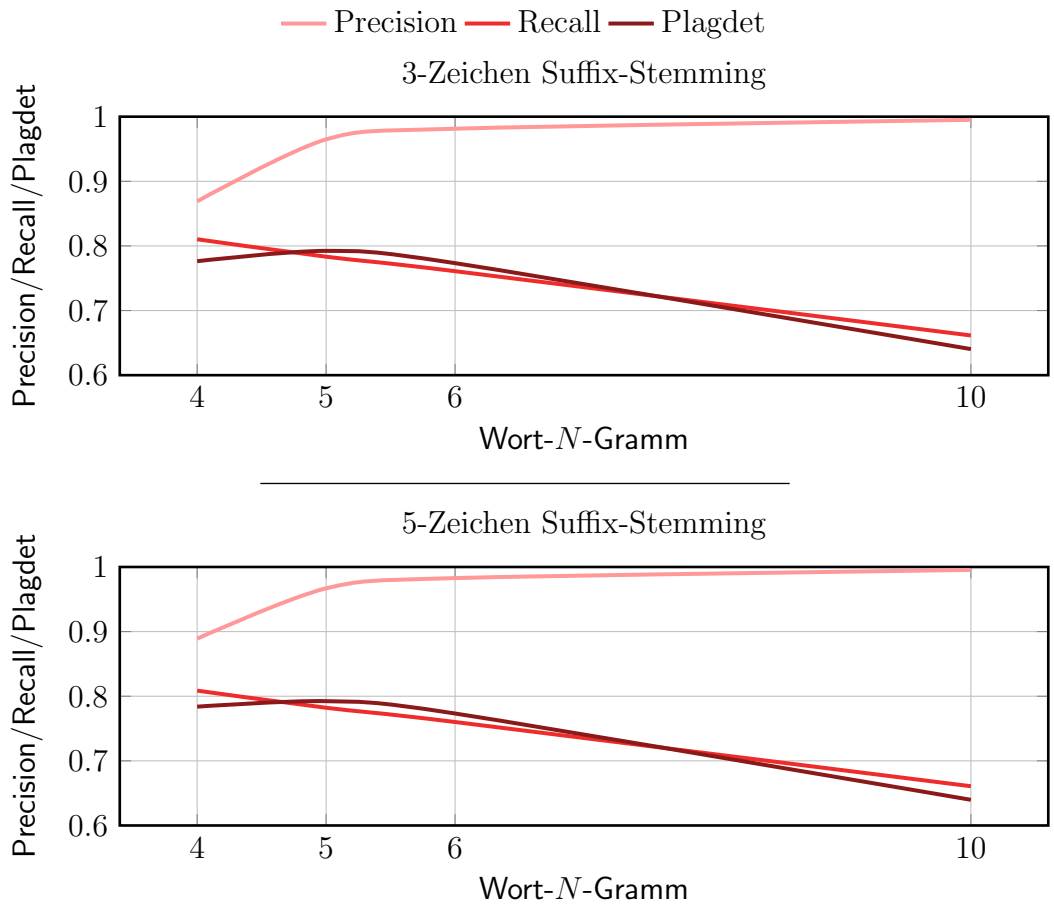


Abbildung 4.3. Chunking-Variationen mit guten Trendwerten bei den beiden verwendeten Präfixlängen. Die Precision steigt bei Wort-5-Grammen höher in der Skala ein, ebenso die anderen beiden Maße. Diese verhalten sich allerdings im Überblick bei beiden Beispielen gleich.

Beide Fälle (Abb. 4.3) sorgen für einen Ausgleich zwischen Recall und Precision, deren Werte durchweg über dem angestrebten Mindestwert von 0,5 liegen. Um die Korrelation weiter zu untersuchen sind detaillierte Experimente auf großen Korpora notwendig. Diesem Experiment haben wir die beiden Chunking-Einstellungen Wort-4-Gramm und Wort-5-Gramm als praktikable Trendwerte entnommen.

Sort.	WGram	Precision	Recall	Gran.	Plagdet	Cases	Det.
Yes	4	0,859	0,810	1,114	0,772	9875	10516
	5	0,963	0,784	1.130	0,792	9875	9288
	6	0,983	0,763	1.156	0,775	9875	9142
No	4	0,893	0,807	1,115	0,784	9875	10109
	5	0,969	0,781	1.131	0,792	9875	9211
	6	0,984	0,759	1.157	0,773	9875	9112

Tabelle 4.6. Die Sortierung der Token hat Auswirkung auf die Menge ähnlicher Token und die Konfidenz einer relevanten Zuordnung der Text-Reuse-Instanzen. Die Daten dieser Tabelle wurden mit einem 5-Zeichen Suffix-Stemming berechnet und entsprechen den gleichen Einstellungen des vorherigen Experiments.

Sortierung der Chunks

Alle bisherigen Experimente wurden ohne alphabetische Sortierung der Token durchgeführt. In diesem Test untersuchten wir die Veränderung der Erkennungsleistung in Bezug auf die Sortierung der Token, die dadurch dieses Vorgehen ähnlicher zueinander werden. Die Sortierung bewirkt eine verhältnismäßig starke Normalisierung im Vergleich zu den vorhergehenden Filtern der Vorverarbeitung. Viele Token mit ähnlichen Wörtern, die sich bisher durch die originale Reihenfolge unterschieden, sind durch die Entfernung dieses Merkmals direkt vergleichbar. Wie die Daten in Tabelle 4.6 zeigen, sinkt mit der Sortierung der Token die Precision durch die zunehmende Vereinheitlichung. Der marginale Unterschied zwischen aktivem und inaktivem Sortieren ist auffällig und in gleichem Maße speziell für die verwendeten anderen Parameterkonstellationen. Der Trend der Parameterausprägungen (Abb. 4.4) legt die Vermutung nahe, dass die Sortierung der Token ein Parameter zur Recall-Steuerung ist. Nach diesem Experiment wurde die Sortierung der Token ebenfalls als variabler Parameter in zukünftigen Experimenten verwendet.

Clusterfilterung

Die Text-Reuse-Analyse basiert auf dem Vergleich von Textfragmenten. Abhängig von der Größe der Textfragmente, tauchen Übereinstimmungen bei Vergleichen dieser Art häufig als zufällige Wiederholungen auf.

Der Clusterfilter entfernt deshalb kleine Cluster anhand eines Schwellwertes, um das Rauschen zufälliger Übereinstimmungen kleiner Textfragmente zu reduzieren. In Abbildung 4.5 sind die Entwicklungen der Leistungswerte bezogen auf veränderte Mindestgrößen für erkannte Cluster dargestellt. Alle Leistungswerte beruhen auf einem festen Wort-5-Gramm Chunking und einer Präfixlänge von 5 Zeichen. Die Resultate spiegeln sich in gleicher Form mit an-

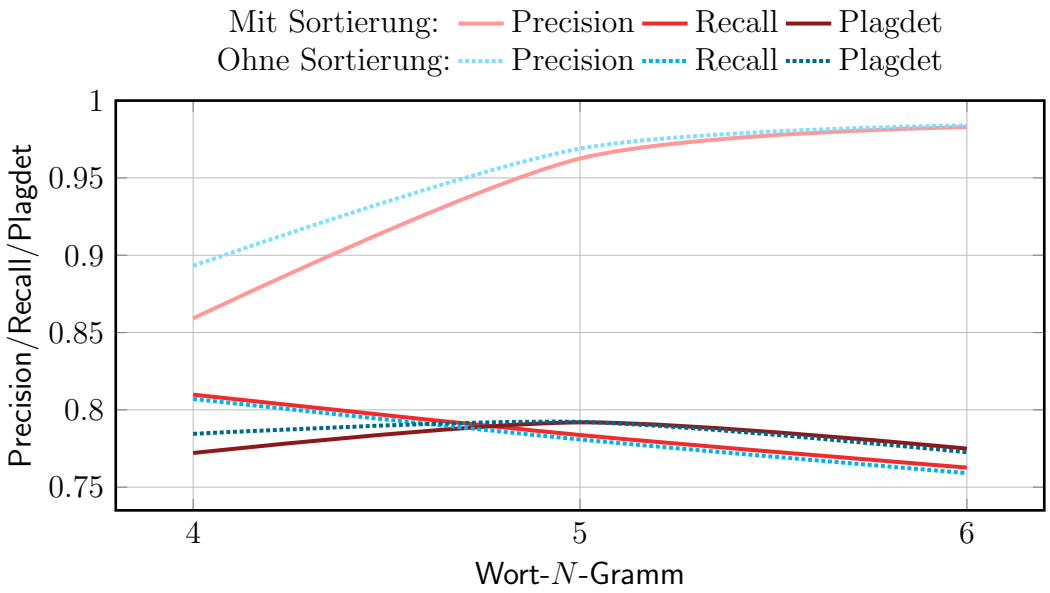


Abbildung 4.4. Die Visualisierung der Leistungswerte zeigt einen deutlichen Unterschied zwischen beiden Varianten, aktive und inaktive Sortierung. Das verwendete 5-Zeichen Suffix-Stemming verhält sich hier gleichwertig zum kleineren 4-Zeichen Suffix-Stemming.

deren Parameterkonstellationen wieder. Deutlich sichtbar ist der Anstieg der Precision ab einer Mindestgröße von 80 Zeichen und die einsetzende Ernüchterung der Recallwerte ab 100 Zeichen. Wir haben uns dafür entschlossen, den Zwischenwert der beiden genannten Größen mit 90 Zeichen als Schwellwert zu wählen. Diese Größe entspricht aus unserer Sicht etwa der Länge von Sätzen. Der Verlust von Textinformationen ist in diesem Fall in zumutbaren Grenzen gehalten. Dennoch bleibt die Möglichkeit bestehen, eine Text-Reuse-Analyse mit relativ kleinen Textfragmenten und guten Leistungswerten durchzuführen.

Clusteranalyse

Die Clusteranalyse des Verfahrens besteht aus zwei in Reihe geschalteten DBScan-Instanzen, die durch den Clusterfilter getrennt sind. Beide Instanzen unterscheiden sich lediglich in der Wahl der Parameter. Insgesamt vier Parameter steuern diese Schritte. Aufgrund der Notwendigkeit des DBScan-Verfahrens die Parameter an die Datenlage anpassen zu müssen, ergibt sich für den Radius der ϵ-Nachbarschaft und für den Schwellwert $MinPts$ des Kernpunktkriteriums ein weiter Möglichkeitenraum. In den folgenden Abbildungen ist ein Ausschnitt der von uns analysierten Einstellungen mit den Leistungswerten des Verfahrens dargestellt. Der Wertebereich wurde durch Annahmen über

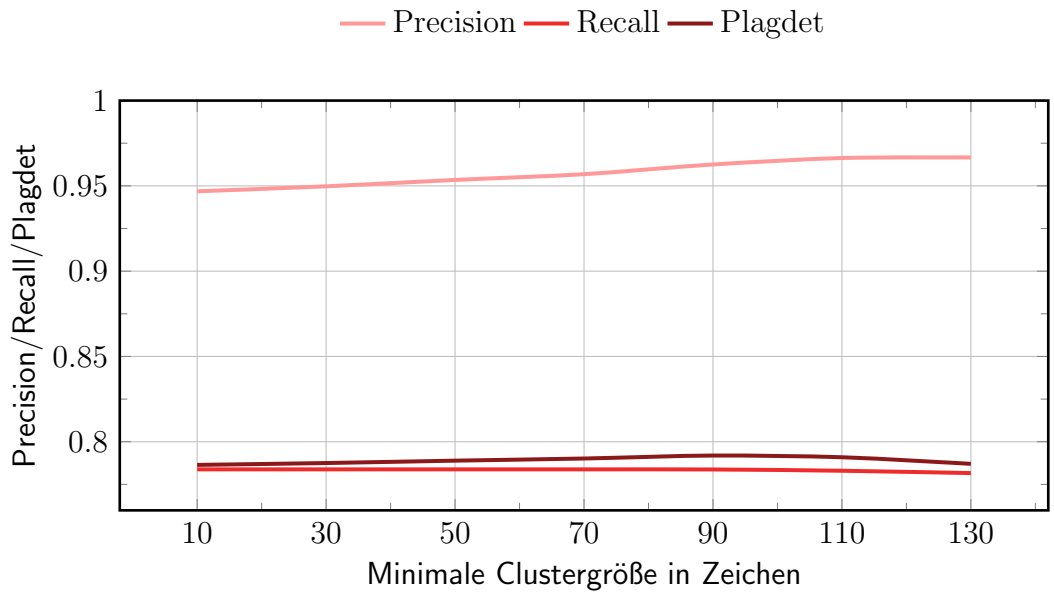


Abbildung 4.5. Der Clusterfilter besitzt überwiegend konstante Leistungswerte. Lediglich innerhalb beschränkter Bereiche zwischen 70 und 110 Zeichen ist eine tendenzielle Verbesserung zu erkennen.

das Verhalten eines lokalen und globalen Alignments im Voraus eingegrenzt.

Der getestete DBScan-Radius der ersten Clustering-Instanz lässt, in Verbindung mit den Leistungswerten (Abb. 4.6), für beide Stemming-Varianten ein verwertbares Maß zwischen 750 und 1500 Zeichen erkennen. Eine Veränderung der Chunklänge bei gleichbleibenden Stemming-Varianten ergibt diesbezüglich keine generellen Unterschiede.

Die weiteren Experimente beziehen sich auf ein Wort-5-Gramm Chunking mit aktiver Sortierung und einer Prefixlänge von 5 Zeichen. Diese Paramtereinstellungen haben sich als stabile Varianten ohne größere Fluktuationen in den Leistungswerten herausgebildet.

Die zweite DBScan-Instanz (Abb. 4.7) wird beeinflusst von den Ergebnissen der ersten Instanz. Wie sich in den Experimenten bestätigt hat, führt die Wahl extrem großer Radien für beide DBScan-Instanzen zu keinem zufriedenstellenden Erfolg. Die Verbindung der beiden Clusteranalysen erlangt dadurch keinen Effekt und die Erkennungsleistung sinkt mit der Precision rapide auch wenn der Recall steigt. Die Radien beider Clusteranalysen unterscheiden sich in den aktuell vorgegebenen Einstellungen jeweils um ein Drittel. Wir haben bei den Experimenten mit dem kleinen Testkorpus, die Veränderungen der Radien in begrenzten Schritten vorgenommen, um Tendenzen abzuschätzen und eine Spezialisierung

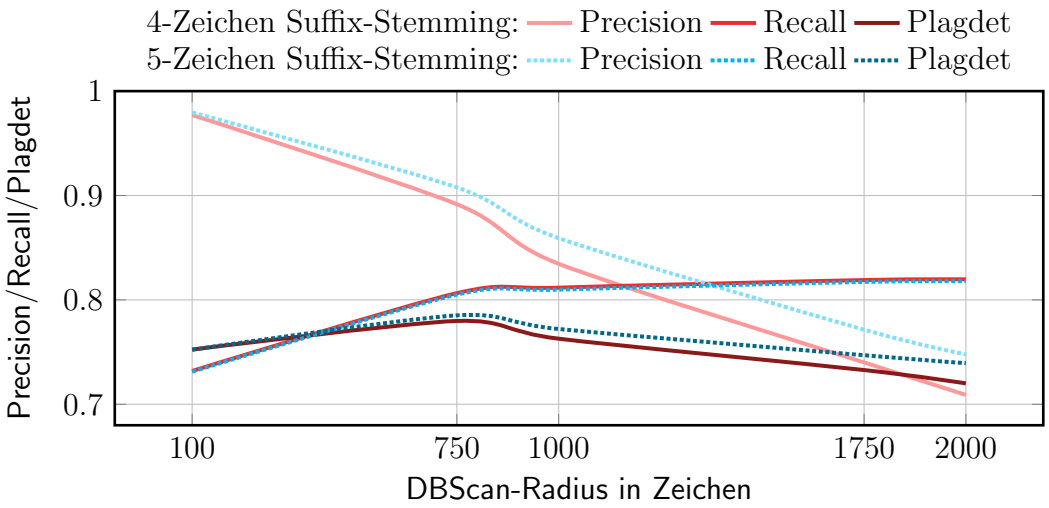


Abbildung 4.6. Die Veränderung des ersten DBScan-Radius in Verbindung mit Wort-4-Gramm Chunking und aktivierter Sortierung. Beide Stemming-Varianten weisen ähnliche Wertentwicklungen auf unterscheiden sich jedoch insbesondere bei der Precision.

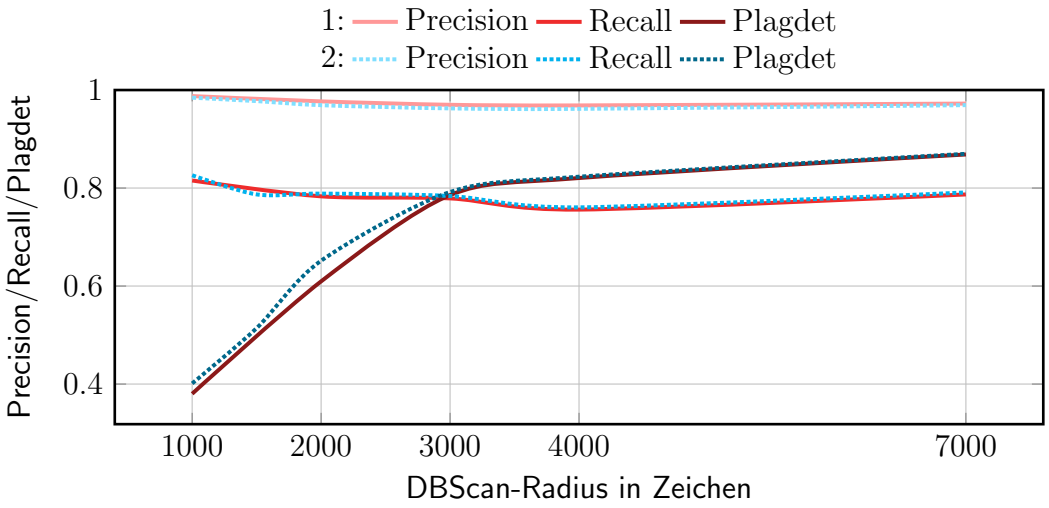


Abbildung 4.7. Die Entwicklung der Leistungswerte bei Veränderung des zweiten DBScan-Radius. Variante 1: mit 750 Zeichen und Variante 2: mit 1000 Zeichen für den ersten DBScan-Radius.

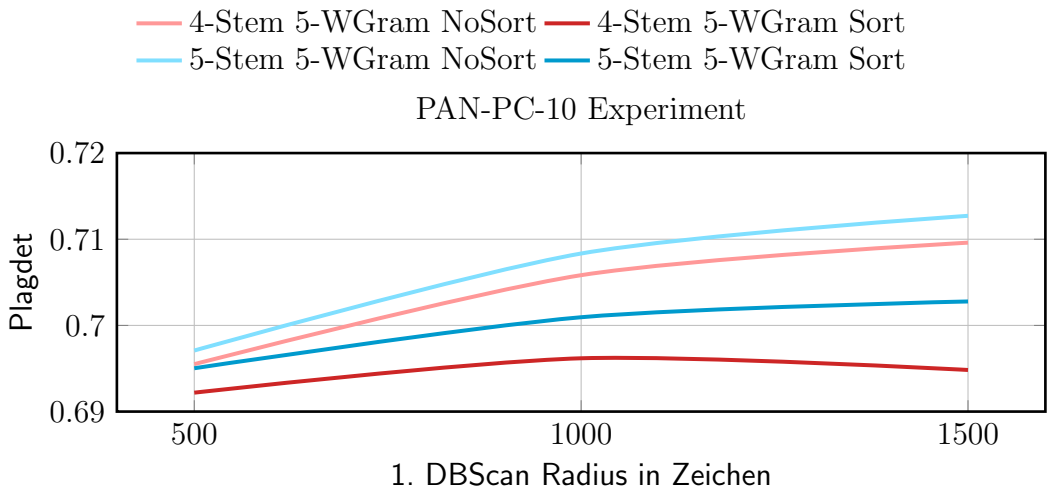


Abbildung 4.8. Entwicklung des PAN-PC-10 Effizienzwertes *Plagdet* mit unterschiedlichen Suffix-Stemming-Varianten. Bei einer Chunklänge von 5 Worten pro Wort-N-Gramm sind die Unterschiede in der Verwendung der Sortierung deutlich zu sehen.

der Parameter für diese Datengrundlage weitestgehend zu vermeiden.

Eine Variation der Parameter für die Clusteranalyse erfordert gefestigte Wertebereiche für die Einstellungen der Textvorverarbeitung. Diesen Möglichkeitenraum haben wir bisher durch stichprobenartige Untersuchungen eingeschränkt.Weitere Experimente müssen speziell die Clusteranalyse evaluieren. Als Kerntechnologie unseres Verfahrens müssen für diese Analysestufe praktikable Parameter gefunden werden. Wir haben in diesem Zusammenhang ausbaufähige Wertebereiche eröffnet, durch die die Qualität und Leistungsfähigkeit unserer Entwicklung einen positiven Trend annimmt. Eine noch nicht zufriedenstellend gelöste Aufgabe ist die vollständige Evaluierung der Parametereinstellungen auf dem PAN-PC.

4.4.3 Experimente mit dem PAN-PC

Mit den erarbeiteten Parametereinstellungen aus den vorangegangenen Experimenten wurden erste Versuche mit dem großen PAN-Plagiatskorpus durchgeführt. Für einen Vergleich mit anderen Entwicklungen wurden die Ergebnisse der PAN-Competition herangezogen. Die verwendeten Korpora stammen aus den Jahren 2010 und 2011. Beide Korpora wurden mit dem Ziel konstruiert, verschiedene Plagiatsfälle in einer kohärenten Testumgebung bereitzustellen [42]. Die Experimente auf dem Korpus unterscheiden sich nicht von dem genannten Verarbeitungsprozess innerhalb der Experimente des Testkorpus.

Rank	Plagdet	Recall	Precision	Granularity	Team
1	0.7971	0.6917	0.9414	1.0006	Kasprzak,Brandejs
	0.7127	0.6895	0.8800	1.1211	Variation 1
2	0.7090	0.6299	0.9055	1.0675	Zou, Long, Ling
3	0.6948	0.7057	0.8417	1.1508	Muhr, Kern, Zechner, Granitzer
4	0.6209	0.4808	0.9085	1.0177	Grozea, Popescu
5	0.6066	0.4768	0.8479	1.0086	Oberreuter, L'Huillier, Ríos, Velásquez
6	0.5851	0.4481	0.8507	1.0044	Torrejón, Ramos
7	0.5191	0.4059	0.7256	1.0039	Pereira, Moreira, Galante
8	0.5093	0.3856	0.7817	1.0195	Palkovskii, Belov, Muzika
9	0.4378	0.2868	0.9561	1.0108	Sobha L., Pattabhi R.K R., Vijay S.R., A. Akilandeswari
10	0.2564	0.3175	0.5062	1.8720	Gottron

Tabelle 4.7. Top-10 der PAN-10 Competition Teilnehmer und Ergebnisse [28]. In den hinterlegten Zeilen sind die Ergebnisse unseres Verfahrens in Bezug auf den Effizienzwert Plagdet (Dunkelgrau) abgetragen.

Die Parametereinstellungen für dieses Experiment sind in Tabelle 4.8 angegeben. Die Entwicklung des Effizienzwertes *plagdet* in der Abbildung 4.8 zeigt den Einfluss der Sortierung der Wort-N-Gramme. Die Zahl der übereinstimmenden Token steigt mit aktiver Sortierung und gleichzeitiger Verringerung der Präfixlängen des Suffix-Stemming. Aus diesem Grund steigt die Menge erkannter Textstellen drastisch an, jedoch wird gleichfalls die Genauigkeit gemindert. Das Verfahren erkennt damit zu viele falsche Text-Reuse-Instanzen zwischen den echten wiederverwendeten Textstellen und fragmentiert dadurch deutlich die Endergebnisse.

Das beste Ergebnis dieses Experiments wird mit den in Tabelle 4.8 angegebenen Einstellungen erreicht. In Tabelle 4.7 sind diese Ergebnisse im Vergleich zu den Top-10 Teilnehmern der PAN-10 Competition aufgelistet.

	Experimente	Beste (Variante 1)
Stoppwortentfernung	Aktiv	
Suffix-Stemming	{4,5}	5
Synonymersetzung	Inaktiv	
Chunking (Wort-N-Gramm)	{4,5}	5
Wort-N-Gramm-Sortierung	In/Aktiv	Inaktiv
DBScan 1 ϵ-Radius	{500,1000,1500}	1500
DBScan 1 $MinPts$	4	
Clusterfilter	90	
DBScan 2 ϵ-Radius	3000	
DBScan 2 $MinPts$	4	

Tabelle 4.8. Parameterausprägungen für die Experimente auf dem PAN-PC-10.

	Experimente	Beste (Variante 1)
Stoppwortentfernung	Aktiv	
Suffix-Stemming	{4,5}	5
Synonymersetzung	Inaktiv	
Chunking (Wort-N-Gramm)	{4,5}	4
Wort-N-Gramm-Sortierung	In/Aktiv	Inaktiv
DBScan 1 ϵ-Radius	{500,1000,1500}	1500
DBScan 1 $MinPts$	4	
Clusterfilter	90	
DBScan 2 ϵ-Radius	3000	
DBScan 2 $MinPts$	4	

Tabelle 4.9. Parameterausprägungen für die Experimente auf dem PAN-PC-11.

Mit dem PAN-PC-11 wurden ebenfalls Experimente durchgeführt, die sich aus den in Tabelle 4.9 aufgelisteten Einstellungen zusammensetzen. Unsere Ergebnisse sind bei diesem Experiment mit verminderter Effizienz im Vergleich zum PAN-PC-10 ausgefallen. Die Evaluierung der Parameter ist durch die Abschätzung der Trendwerte noch nicht vollständig zu stabil verwertbaren Einstellungen für heterogene Daten gelangt. Hinsichtlich des Korpus existieren nun vermehrt Präferenzen zu starker Obfuskierung der wiederverwendeten Textabschnitte. Hoch obfuskierte Textstellen versuchen, reale Plagiate weitestgehend zu simulieren. Dieser Punkt macht sich auch in der Analyse bemerkbar. Die vermehrt eingestreuten Änderungen dünnen die Ballungsbereiche im Dotplot aus und verteilen sie gleichzeitig auf ein größeres Gebiet.

Bezüglich dieser Inhalte ist unser Verfahren durch seinen syntaktischen Vergleich mit den aktuellen Einstellungen noch zu grob. Mit den Ergebnissen innerhalb des Teilnehmerfeldes der PAN-11 Competition (Tabelle 4.10) sind wir allerdings auf einem guten Weg mit ausbaufähigen Grundlagen.

Rank	Plagdet	Recall	Precision	Granularity	Team
1	0.5563430	0.3965569	0.9368736	1.0022487	Grman, Ravas
2	0.4153395	0.3376925	0.8119867	1.2167900	Grozea, Popescu
3	0.3468605	0.2257937	0.9116530	1.0611984	Oberreuter
	0.2893	0.2032	0.5811	1.0576	Variation 1
4	0.2467329	0.1500480	0.7106536	1.0058894	Gillam
5	0.2340035	0.1612845	0.8512947	1.2328923	Torrejón, Martín Ramos
6	0.1990889	0.1618067	0.4541152	1.2949292	Gupta, Singhal, Sameer, Majumder
7	0.1892155	0.1390201	0.4435687	1.1716516	Palkovskii
8	0.0804139	0.0885330	0.2780244	2.1823870	Nawab, Stevenson, Clough

Tabelle 4.10. PAN-11 Competition Teilnehmer und Ergebnisse [42]. Die hinterlegte Zeile zeigt die Leistungswerte unseres Verfahrens im direkten Vergleich für den besten Plagdet-Wert (Dunkelgrau).

Kapitel 5

Schlussbemerkung

Im Rahmen dieser Arbeit wurde ein Verfahren für die Text-Reuse-Analyse entwickelt, das wiederverwendete Textabschnitte in einem paarweisen Dokumentvergleich identifizieren und extrahieren kann. Es abstrahiert identische Textfragmente von der Länge weniger Wörter in einem Punktefeld. Die Identifikation wiederverwendeter Textabschnitte erfolgt durch eine automatische Mustererkennung in diesem Punktefeld. Daraufhin werden mithilfe der erkannten Muster gleiche und ähnliche Textabschnitte extrahiert und einer Weiterverarbeitung zur Verfügung gestellt. Die in dieser Arbeit vorgestellte Text-Reuse-Analyse setzt Verfahren zur Textnormalisierung und automatischen unüberwachten Klassifizierung ein. Diese Technologien besitzen insgesamt zehn Parameter mit variablen Wertebereichen.

Innerhalb dieser Arbeit wurde die vorgestellte Text-Reuse-Analyse entwickelt und die angesprochenen Parameter auf verwertbare Einstellungen untersucht. Das Verfahren besteht aus zwei Stufen, die sich in Vorverarbeitung und Analyse unterteilen. Für die fünf Parameter der Vorverarbeitung wurde eine Sensitivitätsanalyse durchgeführt und die Einstellungen anhand der erbrachten Leistungswerte des Verfahrens untersucht. Die verbleibenden fünf Parameter beziehen sich auf die beiden Clusteranalysen und den Clusterfilter. Für diese Parameter wurden sinnvolle Annahmen getroffen, um mit der Analysestufe dem beabsichtigten Verhalten eines lokalen und globalen Sequence-Alignment nahezukommen. Für beide Clusteranalysen wurde damit der Lösungsraum möglicher Parameterkombinationen eingeschränkt.

Die Analyse mit den Daten des Testkorpus resultierte in Wertebereichen, die bezogen auf heterogene Textsammlungen wie den PAN-PC lediglich Tendenzen widerspiegelten. Die zentralen Parameter bilden mit der Größe der Wort-N-Gramme und der Präfixlängen des Suffix-Stemming in Verbindung mit den beiden Clusteranalysen einen unüberschaubaren Kombinationsbereich. Für die ersten beiden Einstellungen wurden verwendbare Wertebereiche un-

tersucht und als Grundlage für weitere Schritte festgehalten. Die Länge der Wort-N-Gramme kann zwischen vier und sechs Worten variieren. Während dieser Variation kann die Präfixlänge zwischen drei und fünf Zeichen verändert werden. Die endgültigen Ergebnisse dieser Text-Reuse-Analyse sind abhängig von den Einstellungen der Clusteranalysen. Folglich müssen die Parameter des DBScan-Verfahrens auf die Daten eingestellt werden. Innerhalb dieser Arbeit wurden die Parameter auf sinnvolle Voreinstellungen festgelegt. Die erste Clusteranalyse wurde, unter der Annahme lokale Dichtebereiche zu bevorzugen, mit einem kleinen Wertebereich zwischen 500 und 2000 Zählern im Radius untersucht. Darauf folgte die zweite Clusteranalyse mit einem dreifach größeren Radius. Diese Einstellungen wurden, im Rahmen dieser Arbeit, als Richtwerte für das beabsichtigte Wirken auf lokale und globale Bereiche innerhalb der Dotplots angenommen.

Durch die Aufspaltung unseres Verfahrens in zwei Verarbeitungsstufen existieren verschiedene Erweiterungsmöglichkeiten. Die Vorverarbeitungsstufe kann mit zusätzlichen Filtern ausgebaut werden. Diese Möglichkeit wird durch die eingesetzte Filterarchitektur sichergestellt. Gleichfalls ist die Verarbeitungs-Pipeline der Analysestufe nicht festgelegt, kann jedoch als Ausgangsbasis für Erweiterungen dienen.

Weitere Problemstellungen

Die Parameter der Clusteranalyse mit DBScan sind abhängig von den Eingabedaten des Verfahrens. Diese statische Spezialisierung ist noch unflexibel gegenüber der beabsichtigten Variabilität der Eingabedaten. Folglich ist es weiterhin notwendig, diese Einstellungen nach einer Überprüfung der Ergebnisse abhängig von einer spezifischen Text-Reuse-Analyse manuell zu justieren. Die automatische Wahl der DBScan-Parameter in Abhängigkeit der Eingabedaten ist ein Problem, das trotz optimierter DBScan-Implementierungen noch nicht zufriedenstellend gelöst ist.

Die Filter-Pipeline der Textvorverarbeitung enthält zunächst eine grundlegende Menge normalisierender Filterverfahren. Die bisherige Textanalyse ist vorerst ohne die Synonymnormalisierung ausgekommen. Dieses zusätzliche Verfahren erfordert durch seine offensichtliche Sprachabhängigkeit einen erheblich Mehraufwand, der zunächst zurückgestellt wurde. Darunter zählt in erster Linie die Überprüfung zusätzlicher Thesauri in Anlehnung an WordNet.

Um das in dieser Arbeit entwickelte Verfahren auch für mehrsprachige Textdokumente und eine sprachübergreifende Text-Reuse-Analyse einzusetzen, wurden bereits grundlegende Vorbereitungen getroffen.

Literaturverzeichnis

[1] Apache Hadoop Project: *Apache Hadoop*. `http://hadoop.apache.org/`, 29. August 2011.

[2] Apache Lucene Project: *Apache Lucene*. `http://lucene.apache.org/`, 29. August 2011.

[3] Baeza-Yates, Ricardo A. und Berthier Ribeiro-Neto: *Modern Information Retrieval*, Kapitel 1.1. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2011, ISBN 978-0-321-41691-9.

[4] Baeza-Yates, Ricardo A. und Berthier Ribeiro-Neto: *Modern Information Retrieval*, Kapitel 1.2. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2011, ISBN 978-0-321-41691-9.

[5] Baeza-Yates, Ricardo A. und Berthier Ribeiro-Neto: *Modern Information Retrieval*, Kapitel 6.5. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2011, ISBN 978-0-321-41691-9.

[6] Baeza-Yates, Ricardo A. und Berthier Ribeiro-Neto: *Modern Information Retrieval*, Kapitel 6.5.3. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2011, ISBN 978-0-321-41691-9.

[7] Baeza-Yates, Ricardo A. und Berthier Ribeiro-Neto: *Modern Information Retrieval*, Kapitel 6.6. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2011, ISBN 978-0-321-41691-9.

[8] Barrón-Cedeño, Alberto, Martin Potthast, Paolo Rosso, Benno Stein und Andreas Eiselt: *Corpus and Evaluation Measures for Automatic Plagiarism Detection*. In: Calzolari, Nicoletta, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner und Daniel Tapias (Herausgeber): *7th Conference on International Language Resources and Evaluation (LREC 10)*. European Language Resources Association (ELRA), Mai 2010, ISBN 2-9517408-6-7.

[9] Bentley, J. L.: *Multidimensional binary search trees used for associative searching.* Communications of the ACM, 18(9):509–517, September 1975.

[10] Brown, Daniel G.: *A survey of seeding for sequence alignment*, 2007.

[11] Church, Kenneth Ward und Jonathan Isaac Helfman: *Dotplot: a Program for Exploring Self-Similarity in Millions of Lines of Text and Code*, 1993.

[12] Clough, P.: *Measuring text reuse in a journalistic domain.* In: *Conference Proceedings of the 4th Annual CLUK Colloquium, University of Sheffield, Sheffield, UK*, 2001.

[13] Cooper, William S.: *A definition of relevance for information retrieval.* Information Storage and Retrieval, 7(1):19–37, 1971.

[14] Ester, Martin, Hans Peter Kriegel, Jörg Sander und Xiaowei Xu: *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* In: *KDD*, Seiten 226–231, 1996.

[15] Frakes, William B. und Ricardo A. Baeza-Yates: *Information Retrieval: Data Structures & Algorithms*, Kapitel 8. Prentice-Hall, 1992, ISBN 0-13-463837-9.

[16] Frakes, William B. und Ricardo A. Baeza-Yates: *Information Retrieval: Data Structures & Algorithms*, Kapitel 16.1.1. Prentice-Hall, 1992, ISBN 0-13-463837-9.

[17] Frakes, William B. und Ricardo A. Baeza-Yates: *Information Retrieval: Data Structures & Algorithms*, Kapitel 16.1.2. Prentice-Hall, 1992, ISBN 0-13-463837-9.

[18] Gibbs, A. J. und G. A. McIntyre: *The diagram: A method for comparing sequences. Its uses with amino acids and nucleotide sequences.* Eur. J. Biochem., 16:1–11, 1970.

[19] Grozea, C., C. Gehl und M. Popescu: *ENCOPLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection.* In: *3rd PAN WORKSHOP. UNCOVERING PLAGIARISM, AUTHORSHIP AND SOCIAL SOFTWARE MISUSE*, Seite 10, 2009.

[20] Helfman, Jonathan: *Dotplot Patterns: A Literal Look at Pattern Languages.* Theory and Practice of Object Sytems, 2(1):31–41, 1996.

[21] Johann Bachner: *Clusteranalyse: Anwendungsorientierte Einführung in Klassifikationsverfahren By Johann Bacher*, Kapitel 1. Oldenbourg Wissenschaftsverlag, 2010.

[22] Jones, N. C. und P. A. Pevzner: *An Introduction to Bioinformatics Algorithms*. MIT Press, 2004.

[23] Krafft, Martin F., Paul Harris und Sylvain Bougerel: *C++ template container implementation of kd-tree sorting*. `http://libkdtree.alioth.debian.org/`, 29. August 2011.

[24] Kumar, J Prasanna und P Govindarajulu: *Duplicate and Near Duplicate Documents Detection : A Review*. European Journal of Scientific Research, 32(4):514–527, 2009.

[25] Leskovec, J., L. Backstrom und J. Kleinberg: *Meme-tracking and the dynamics of the news cycle*. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, Seiten 497–506. ACM, 2009.

[26] Mount, David M.: *Bioinformatics - sequence and genome analysis (2. ed.)*, Kapitel 3, Seiten I–XII, 1–692. Cold Spring Harbor Laboratory Press, 2004, ISBN 978-0-87969-687-0.

[27] Mount, David M.: *Bioinformatics - sequence and genome analysis (2. ed.)*, Kapitel 4, Seiten I–XII, 1–692. Cold Spring Harbor Laboratory Press, 2004, ISBN 978-0-87969-687-0.

[28] Potthast, Martin, Alberto Barrón-Cedeño, Andreas Eiselt, Benno Stein und Paolo Rosso: *Overview of the 2nd International Competition on Plagiarism Detection*. In: Braschler, Martin und Donna Harman (Herausgeber): *Notebook Papers of CLEF 10 LABs and Workshops*, September 2010, ISBN 978-88-904810-0-0.

[29] Potthast, Martin, Benno Stein, Alberto Barrón-Cedeño und Paolo Rosso: *An Evaluation Framework for Plagiarism Detection*. In: Huang, Chu Ren und Dan Jurafsky (Herausgeber): *23rd International Conference on Computational Linguistics (COLING 10)*, Seiten 997–1005, Stroudsburg, PA, USA, August 2010. Association for Computational Linguistics.

[30] Potthast, Martin, Benno Stein, Andreas Eiselt, Alberto Barrón-Cedeño und Paolo Rosso: *Overview of the 1st International Competition on Plagiarism Detection*. In: Stein, Benno, Paolo Rosso, Efstathios Stamatatos, Moshe Koppel und Eneko Agirre (Herausgeber): *SEPLN 09 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09)*, Seiten 1–9. CEUR-WS.org, September 2009.

[31] Priestley, M.: *DITA XML: A reuse by reference architecture for technical documentation.* In: *Proceedings of the 19th annual international conference on Computer documentation*, Seite 156. ACM, 2001.

[32] Princeton University: *WordNet - A lexical database for English.* `http://wordnet.princeton.edu/`, 29. August 2011.

[33] Shannon, C. E.: *Prediction and entropy of printed English.* Bell Systems Technical Journal, 30:50–64, 1951.

[34] Sigurd, Bengt, Mats Eeg-Olofsson und Joost Van Weijer: *Word length, sentence length and frequency – Zipf revisited.* Studia Linguistica, 58(1):37–52, 2004, ISSN 1467-9582.

[35] Stein, Benno: *Fuzzy-Fingerprints for Text-Based Information Retrieval.* In: Tochtermann, Klaus und Hermann Maurer (Herausgeber): *Proceedings of the 5th International Conference on Knowledge Management (I-KNOW 05), Graz, Austria*, Journal of Universal Computer Science, Seiten 572–579. Know-Center, Juli 2005.

[36] Stein, Benno: *Principles of Hash-based Text Retrieval.* In: Clarke, Charles, Norbert Fuhr, Noriko Kando, Wessel Kraaij und Arjen P. de Vries (Herausgeber): *30th Annual International ACM SIGIR Conference (SIGIR 07)*, Seiten 527–534. ACM, Juli 2007, ISBN 987-1-59593-597-7.

[37] Stein, Benno: *Lectures on cluster analysis.* Slides at Bauhaus-Universität Weimar, 2011.

[38] Stein, Benno: *Lectures on information retrieval.* Slides at Bauhaus-Universität Weimar, 2011.

[39] Stein, Benno und Michael Busch: *Density-based Cluster Algorithms in Low-dimensional and High-dimensional Applications.* In: Stein, Benno und Sven Meyer zu Eißen (Herausgeber): *Second International Workshop on Text-Based Information Retrieval (TIR 05), Koblenz, Germany*, Fachberichte Informatik, Seiten 45–56. University of Koblenz-Landau, Germany, September 2005.

[40] Stein, Benno und Sven Meyer zu Eißen: *Document Categorization with MajorClust.* In: Basu, Amit und Soumitra Dutta (Herausgeber): *12th Workshop on Information Technology and Systems (WITS 02)*, Seiten 91–96. Technical University of Barcelona, Dezember 2002.

[41] Stein, Benno, Sven Meyer zu Eißen und Martin Potthast: *Strategies for Retrieving Plagiarized Documents*. In: Clarke, Charles, Norbert Fuhr, Noriko Kando, Wessel Kraaij und Arjen P. de Vries (Herausgeber): *30th Annual International ACM SIGIR Conference (SIGIR 07)*, Seiten 825–826, New York, Juli 2007. ACM, ISBN 987-1-59593-597-7.

[42] Stein, Benno, Martin Potthast, Alberto Barrón-Cedeño, Paolo Rosso, Efstathios Stamatatos und Moshe Koppel: *Fourth International Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 10)*. SIGIR Forum, 45(1):45–48, Juni 2011, ISSN 0163-5840.

[43] Webis Group `http://www.webis.de/`: *AITools*. `http://www.aitools.de/`, 29. August 2011.

[44] Xu, R. und D. Wunsch: *Survey of clustering algorithms*. IEEE Transactions on Neural Networks, 16(3):645–678, Mai 2005.