**Bestimmung der Dokumentenähnlichkeit basierend auf Bayesscher Statistik für eine Big-Data Information Retrieval Lösung**

Master-Thesis im

Studiengang Wirtschaftsinformatik

Fakultät Informatik

Hochschule Reutlingen

Alteburgstraße 150

72762 Reutlingen

Eingereicht von

Elisabeth Agnes Mpessa Enangue

Pestalozzistr.29

72762 Reutlingen

Matrikel Nr.: 713115

Betreuer/Prüfer

Prof Dr.-Ing. Christian Decker

Prof. Dr. rer pol. Dietmar Bönke  
Steve Strauch

Tag der Abgabe: 31.Juli 2018

Inhalt

[Zusammenfassung iii](#_Toc519013220)

[Abstract (Englisch) iv](#_Toc519013221)

[1 Einleitung 1](#_Toc519013222)

[1.1 Motivierendes Beispiel 2](#_Toc519013223)

[1.2 Problemstellung 3](#_Toc519013224)

[1.3 Definitionen und Konventionen 3](#_Toc519013225)

[1.4 Aufbau der Arbeit 4](#_Toc519013226)

[2 Grundlagen 5](#_Toc519013227)

[2.1 Informationsrückgewinnung 5](#_Toc519013228)

[2.2 Bayessche Statistik 5](#_Toc519013229)

[2.3 BigData4Biz 5](#_Toc519013230)

[2.3.1 Vorstellung der Firma dibuco GmbH 5](#_Toc519013231)

[2.3.2 Vorstellung der Software BigData4Biz 5](#_Toc519013232)

[2.3.3 Funktionelle Anforderungen 6](#_Toc519013233)

[2.3.4 Nicht funktionelle Anforderungen 6](#_Toc519013234)

[2.3.5 Plattformanforderungen 7](#_Toc519013235)

[2.3.6 Komponentenanforderungen 8](#_Toc519013236)

[2.3.7 Anforderungen am Ähnlichkeitsalgorithmus 12](#_Toc519013237)

[2.3.8 High-Level Architektur 13](#_Toc519013238)

[2.4 Wichtigen Begriffen und Definitionen aus der Ähnlichkeitsbestimmung von Dokumenten 16](#_Toc519013239)

[2.4.1 Herangehensweise bei der Untersuchung von Dokumentähnlichkeit 16](#_Toc519013240)

[2.4.2 Definition von wichtigen Begriffe aus der Ähnlichkeitsbestimmung von Dokumenten 18](#_Toc519013241)

[2.5 Ansätze zur Dokumentähnlichkeitsbestimmung 19](#_Toc519013242)

[2.5.1 Einführung von BayesLSH und BayesLSH-Lite 19](#_Toc519013243)

[2.5.2 Naive Bayes 19](#_Toc519013244)

[2.5.3 Pairwise Compare 19](#_Toc519013245)

[2.6 Vergleich von BayesLSH und Pairwise Compare 20](#_Toc519013246)

[2.7 Die Programmiersprache R 20](#_Toc519013247)

[3 Verwandte Arbeiten 22](#_Toc519013248)

[4 Spezifikation und Design 23](#_Toc519013249)

[4.1 Spezifikation 23](#_Toc519013250)

[4.1.1 Aufgabenstellung 23](#_Toc519013251)

[4.1.2 Beschreibung der Schnittstelle 24](#_Toc519013252)

[4.1.3 Funktionale Anforderungen an das BayesLSH-Lite Algorithmus 25](#_Toc519013253)

[4.1.4 Nicht funktionale Anforderungen an das BayesLSH-Lite Algorithmus 26](#_Toc519013254)

[4.2 Design 27](#_Toc519013255)

[5 Implementierung 30](#_Toc519013256)

[5.1 Die ausgewählte Programmiersprache: R 30](#_Toc519013257)

[5.2 Wichtige Pakete 30](#_Toc519013258)

[5.3 Der Algorithmus BayesLSH-Lite 31](#_Toc519013259)

[5.3.1 Der Code 32](#_Toc519013260)

[5.4 Der Pairwise Compare Algorithmus zur Dokumentähnlichkeitsbestimmung 34](#_Toc519013261)

[5.4.1 Der Code 34](#_Toc519013262)

[6 Evaluation 35](#_Toc519013263)

[6.1 Versuchsaufbau Für der BayesLSH-Lite Algorithmus 35](#_Toc519013264)

[6.1.1 Der Rechner 35](#_Toc519013265)

[6.1.2 Die Daten 36](#_Toc519013266)

[6.1.3 Die Ergebnisse 37](#_Toc519013267)

[6.2 Versuchsaufbau Für der Pairwise Compare Algorithmus 39](#_Toc519013268)

[6.2.1 Die Daten 39](#_Toc519013269)

[6.2.2 Die Ergebnisse 39](#_Toc519013270)

[6.3 Vergleich von beiden Algorithmen 40](#_Toc519013271)

[6.3.1 Die Leistung 41](#_Toc519013272)

[6.3.2 Die Laufzeit 42](#_Toc519013273)

[6.3.3 Die Effizienz 42](#_Toc519013274)

[6.3.4 Die Skalierbarkeit 42](#_Toc519013275)

[6.4 Diskussion der Ergebnisse von beiden Algorithmen 43](#_Toc519013276)

[7. Zusammenfassung und Ausblick 45](#_Toc519013277)

[Abbildungsverzeichnis 46](#_Toc519013278)

[Tabellenverzeichnis 47](#_Toc519013279)

[Quellenverzeichnis 48](#_Toc519013280)

[Index 50](#_Toc519013281)

[Sperrvermerk 51](#_Toc519013282)

[Erklärung 52](#_Toc519013283)

# Zusammenfassung

Die Firma dibuco GmbH entwickelt eine Information Retrieval Middleware-Lösung deren Namen BigData4Biz ist. Die Middleware-Lösung sollte in der Lage sein, Ähnlichkeit zwischen Dokumenten zu bestimmen.

Das Ziel dieser vorliegenden Arbeit ist es Ansätze basierend auf Bayesscher Statistik zu untersuchen für eine Big Data Information Retrieval Lösung (BigData4Biz), die es ermöglichen die Ähnlichkeit zwischen Dokumenten zu bestimmen. Einen von diesen Ansätzen (BayesLSH-Lite) und seinem kollegialen Algorithmus (Pairwise Compare), der nicht auf Bayesscher Statistik basiert, wurden ausgewählt, implementiert und anschließend evaluiert. Der BayesLSH-Lite Algorithmus hat als Basis das Locality Sensitive Hashing (LSH), bzw. die Form davon, nämlich Minhash. Bei Minhash wird eine Familie zufälligen Hashfunktion benutzt um einzelne Dokumente in Hash umzuwandeln. Dabei werden nur von jeder eindeutigen Hashfunktion erzeugte minimalen Werte beibehaltet. Die Ähnlichkeitsmessung von zwei Minhashes erfolgt dann Mithilfe der Jaccard Ähnlichkeit. Der BayesLSH-Lite Algorithmus vergleicht eine Menge von Kandidatenpaaren, die unter Minhashes Formen Tauchen, berechnet ihren Jaccard Ähnlichkeiten und unter der Bedingung, dass das Posterior (Satz von Bayes) kleiner ist als einen gegebenen Abrufsparameter, liefert die Liste der ähnlichen Dokumente zurück mit der entsprechenden Jaccard Ähnlichkeit.

Anhand von diesem o. g Algorithmen wurde die Dokumentähnlichkeit zwischen mehreren Dokumenten bestimmt. Darüber hinaus wurde einen Mehrwert erzielt, da es den Nutzer ermöglicht, schnell nach bestimmte Informationen zu suchen oder anhand eines Dokuments wissen zu können, welche anderen Dokumente fast das gleiche oder noch besser das gleiche Thema bespricht. Ein Vergleich dieser beiden Algorithmen hilft dabei eine Empfehlung bezüglich des BayesLSH-Lite Algorithmus zu machen.

# Abstract (Englisch)

# Einleitung

Angesicht des operativen Tagesgeschäfts und der Geschäftsentwicklung nach Jahren ist die Datenflut im unternehmerischen Umfeld nicht zu stoppen. Die Speicherung von Großer Datenmengen zur Auswertung und Verarbeitung erfolgt in mehreren konventionellen Datenbanken. Big Data umfasst alle Daten, die sowohl zur Verarbeitung als auch zur Analyse oder Auswertung versehen sind und deren Menge konsiderabel wächst. Darüber hinaus ist Big Data charakterisiert durch das große Volum an Daten, die Geschwindigkeit der Datenentstehung, die Vielfalt an Datenquellen und Datenformen sowie die Werte der Daten. Die Speicherung von großen Datenvolumen in Datenquellen kann jedoch Schwierigkeiten bereiten, bzw. wenn diese auf mehreren physischen Maschinen erfolgt [FAM16]. Es wird dann für die Datenanalyse in Big Data, Data Lake eingesetzt.

Unter Data Lake wird einen sehr großen Speicher verstanden, der die Daten in ihrem ursprünglichen Rohformat enthält. Das große Datenvolum und die bestehenden strukturierte, semi-strukturierte oder noch unstrukturierte Daten in Data Lake können das Finden jeglicher Daten Mühe und Zeitaufwand für die Mitarbeiter eines Unternehmens bereiten. Die traditionelle Suche und unscharfe Informationsrückgewinnung bieten sich an um dieses Problem zu lösen. Diese sind aber nicht optimal, da sie nur als Basis vorhandenes Wissen haben. Die Strukturierung der Information nach Inhalt und Bedeutung stellt sich als notwendig. Die Einstellung einer Softwarelösung zum Vergleich auf Inhalt von unterschiedlichen Daten aus einem Data Lake ist daher denkbar.

Eine Information Retrieval Middleware-Lösung deren Namen BigData4Biz ist, wird von der Firma dibuco GmbH entwickelt. Diese Softwarelösung beruht auf einem Konzept der mehrdimensionalen Strukturierung eines Data Lakes unter Verwendung verschiedener Begriffe der Datenähnlichkeit. Unter Begriffe der Datenähnlichkeit werden verstanden, die in den Daten enthaltene Begriffe, die als Basis für den Datenvergleich genommen werden. Die Datenähnlichkeit kann entweder geschäftsbezogen oder Geschäft agnostisch sein und beruht auf linguistische Aspekte, die berechnet werden unter anderem mit Berücksichtigung der Begriffsfrequenz und -bedeutung [DIB18]. Die Berücksichtigung der Begriffsfrequenz und -bedeutung hilft dabei Informationen im Data Lake zu gewinnen und zu entdecken. Durch die Bestimmung der Datenähnlichkeit werden relevante und neue Informationen im Data Lake entdeckt.

Eine Programmierschnittstelle ermöglicht es Abfragen in einer GUI zu formulieren um die Informationen zu gewinnen und zu entdecken. Angesichts der großen Menge eine Datensuche mit Benutzung von Sätzen und Schlüsselwörter für eine Abfrage ist unmöglich.

Im Kontext von großen Datenmengen, die immer mehr wachsen, ist die Skalierbarkeit sehr wichtig. Die vorliegende Arbeit setzt sich an diesem Punkt an und untersucht sowohl die unterschiedlichen Ansätze zur Ähnlichkeitsbestimmung, die auf Bayesscher Statistik basieren als auch Ihre Skalierbarkeit. Ziel ist es der bestmögliche Ansatz zur Dokumentähnlichkeit zu finden um eine optimale und adaptierte Datensuche in BigData4Biz zu ermöglichen.

Es werden folgend ein motivierendes Beispiel, die Problemstellung, Definitionen und Konventionen sowie der Aufbau dieser Arbeit genannt.

## Motivierendes Beispiel

In diesem Kapitel wird ein motivierendes Szenario beschrieben, welches als Basis genommen wird um das Ergebnis dieser vorliegenden Arbeit zu prüfen. Dieses Szenario basiert auf die Funktionsweise der Plattform BigData4Biz. Das Szenario basiert auf einen Client/Server Modell zur Dokumentähnlichkeitsbestimmung.

Dokumente eines dibuco Kunde können auf unterschiedliche Speicher gespeichert und auf die Plattform BigData4Biz aufgerufen werden. In diesem Szenario dient die Kunden Seite als Client und die BigData4Biz Plattform als Server. Der Kunde gibt eine Abfrage in Form von Phrasen oder Textausschnitte auf die Abfrage-API von BigData4Biz. Bevor der Dokumentenvergleich stattfindet, müssen erstmal alle Dokumente vom Kunden gesammelt werden. Die auf den Kundespeicher existierende Übertragungsschnittstelle und Agenten ermöglichen es Daten in Entitäten umzuwandeln, um diese später in BigData4Biz nutzen zu können zur Ähnlichkeitsbestimmung. Agenten können JSON-, CSV-, oder noch PDF-Formate sein. Die Entitäten können dann durch die Extract-Transform-Load(ETL)-Methode auf die Plattform BigData4Biz übertragen werden. Sobald Die Entitäten auf der Plattform BigData4Biz sind, werden die verarbeitet, um zu den dabei bestehenden Algorithmen zu passen. Die verarbeiteten Entitäten können dann untersucht werden bei den Ähnlichkeitsalgorithmen.

## Problemstellung

Der Aufwand bei der Bestimmung der Dokumentähnlichkeit für mehrere Quellen ist in der Regel mit hohem Zeitaufwand verbunden. Vor allem die Bestimmung der Ähnlichkeit in einem Haufen von Daten, die sich entweder im Internet befinden oder in gegebenen Datenspeicher. Hier setzt eine Informationsrückgewinnungslösung an. Diese versucht die Bestimmung der Ähnlichkeit zwischen Dokumente durchzuführen. Dafür gibt es mittlerweile eine Vielzahl an Ansätze. Je mehr die Übereinstimmung sein soll, desto höher wird der Aufwand sein. Darüber hinaus ergibt sich, dass für die Bestimmung der Dokumentähnlichkeit zwei sich gegenseitig gegenüberstehende Herausforderungen beeinflussen, nämlich der Bestimmungsaufwand und den Wunsch Dokumente mit der höchsten Übereinstimmung im gesamten Data Lake zu finden. Da dadurch die Nutzer zufrieden sein können. Außerdem kann es passieren, dass Dokumente nicht unbedingt zutreffen, obwohl die als ähnlich bestimmt wurden. Faktoren wie den Kontext, die Semantik sowie die Bedeutung müssen berücksichtigt werden. Um dieses Problem zu lösen, soll einen bestimmten Ansatz basierend auf Bayesscher Statistik ausgewählt werden, der diese Faktoren berücksichtigt. Die vorliegende Arbeit untersucht auf Bayesscher Statistik basierende Ansätze zur Dokumentähnlichkeitsbestimmung.

## Definitionen und Konventionen

In diesem Abschnitt wird die volle Bedeutung von Abkürzungen gegeben.

**Liste von Abkürzungen**

Die folgende Liste besteht aus Abkürzungen, die in dieser Arbeit benutzt werden

**LSH** Locality Sensitive Hashing

**CSV** Comma-Separated Values

**XML** Extended Markup Language

**RDB** relational Database

**XSLT** extensible Stylesheet Language Transformation

**TF-IDF**  term frequency–inverse document frequency

**ETL**  Extract-Transform-Load

**IDE** integrated development environment

**KNIME** Konstanz Information Miner

**MIT** Massachusetts Institute of Technology

**Tf** term frequency

**Idf** Inverse Document Frequency

**JS**  JavaScript

**ID** Identifikationsnummer

**UML** Unified Modeling Language

## Aufbau der Arbeit

Die vorliegende Arbeit besteht aus sieben Kapiteln. Das Kapitel 1 befasst sich mit der Einleitung zum Thema dieser Arbeit, wo das Motivierendes Beispiel, die Problemstellung, Definitionen und Konventionen sowie der Aufbau der Arbeit erläutert werden.

Das Kapitel 2 behandelt die Grundlagen, die Informationsrückgewinnung, die Bayessche Statistik, BigData4Biz, wichtigen Begriffen und Definitionen aus der Ähnlichkeitsbestimmung von Dokumenten, Ansätze zur Dokumentähnlichkeitsbestimmung, die Programmiersprache R, und den Vergleich zwischen ausgewählten Algorithmen erläutert werden.

Das Kapitel 3 stellt Verwandte Arbeiten vor, wo es eine Abgrenzung zu dieser Masterarbeit Thema gegeben wird.

Im Kapitel 4 werden die Spezifikation und da Design von zwei Algorithmen vorgestellt.

Im Kapitel 5 wird die Implementierung von ausgewählten Ansätze erläutert.

Im Kapitel 6 wird die Skalierbarkeit der realisierten Ansätze evaluiert durch die Benutzung von ausgewählten Probedaten sowie die Fähigkeiten und Grenzen dieser Ansätze in einer Diskussion erläutert.

Die vorliegende Arbeit wird mit dem Kapitel 7 durch eine Zusammenfassung und einen Ausblick abgeschlossen.

Im folgende Kapitel werden Grundlagen zum besseren Verständnis dieser Arbeit gegeben.

# Grundlagen

Es werden in diesem Kapitel verschiedene Themen behandelt. Erstens die Informationsrückgewinnung, zweitens die Bayessche Statistik, drittens BigData4Biz, viertens wichtige Begriffe aus der Ähnlichkeitsbestimmung von Dokumenten zuletzt die Ansätze zur Dokumentähnlichkeitsbestimmung und anschließend die Programmiersprache R.

## Informationsrückgewinnung

## Bayessche Statistik

## BigData4Biz

BigData4Biz ist ein Produkt der Firma dibuco GmbH. Wer die Firma dibuco ist und was BigData4Biz ist wird im Laufe dieses Unterkapitel erläutert.

### Vorstellung der Firma dibuco GmbH

The Digital Business Company (Kurz: dibuco) ist eine im Jahr 2015 gegründete IT Firma, die Lösungen zur Unterstützung von Unternehmen bei der digitalen Transformation entwickelt und ca. 30 Mitarbeiter beschäftigt. Die Firma dibuco befindet sich in Stuttgart und arbeitet in Partnerschaft mit Hochschulen, renommierten Experten und Forschungsinstituten. Diese bietet unter anderem Strategieberatung, IT-Beratung, IT-Implementierung sowie Coaching in Form von Schulungen für den IT-Betrieb an.

Ein Produkt von dibuco GmbH, worauf sich diese Masterarbeit bezieht, wird folgend vorgestellt.

### Vorstellung der Software BigData4Biz

BigData4Biz wird von der Firma Dibuco GmbH seit April 2017 entwickelt. Diese ist für die Informationsrückgewinnung versehen und ermöglicht es den Nutzer sowohl relevante Informationen zu erkennen als auch gebrauchte Informationen zu gewinnen in einem Data Lake. Unter Annahme, dass die Benutzer ganz unbewusst sind, was die Anzahl und die Art der Informationen und Daten angeht, BD4B bittet die Option für Benutzer bekannte Informationen zu sehen und abzufragen, sodass alternative und ähnliche Informationen als Ergebnisse angezeigt werden, so nach Relevanzgrad und mit einer immer engen Einschränkung bei der Suche. BigData4Biz sollte zukünftig die Möglichkeit anbieten, Ähnliche Dokumente zu bestimmen auf Basis von Ansätze, die auf Bayesscher Statistik basieren. Um dies zu ermöglichen sollten erstmal Anforderungen an diese Softwarelösung erläutert werden.

### Funktionelle Anforderungen

Entitäten entsprechen strukturierte Geschäftsobjekte von Metadaten und Eigenschaften, die assoziiert sind mit weiteren technischen Attributen. Im Falle von gemeinsamen Formate müssen einige Anforderungen erfüllt werden: die Eigenschaftsnamen müssen so vergeben werden, dass die genau gleich sind wie die entsprechenden Werten in der Datenquelle. Außerdem die Lokalisierung des ursprünglichen Werts in den Daten einer Datenquelle (Datensatz, Datei usw.) aus dem Namen der Eigenschaft müsste möglich sein.

Was Entitätstechnischen Attributen angeht, die technische Attributen müssen vom Entitätsmodell abgetrennt werden um erstens sowohl eine Reduzierung der Komplexität zu erzielen als auch der Geschäftsteil vom internen technischen Teil der gesamten Entitätsdaten zu unterscheiden und zweitens die Behandlung der Persistenz technischer Attribute getrennt von der Entität zu erzielen.

Die Werte von relationalen Attribute müssen eindeutig sein für alle Entitäten um eine genaue Zuordnung von den relationalen Services zu erzielen [DIB18].

Was die Beschaffenheit angeht, der gegenseitige Überlauf von zwei schnell auftretende Entitätsaktualisierung oder -löschung in der Lastverarbeitung muss vermieden werden. Die Vergabe von Ladezeitstempel dient zur Vermeidung solcher Situationen. Im Falle von mehreren Instanzen desselben Agenten für Lastausgleichs- oder Hochverfügbarkeitsgründe muss dann garantiert werden, dass eine zeitlich bestmögliche Synchronisierung dieser Agenten durchgeführt wird. Außerdem muss die Anwendung einer Entitätsaktualisierung auf eine zuvor gelöschte Entität vermieden werden.

Nachdem die funktionellen Anforderungen erläutert wurden, gilt es zunächst die nicht funktionellen Anforderungen an BigData4Biz zu nennen.

### Nicht funktionelle Anforderungen

Unter den funktionellen Anforderungen können Eigenschaften wie Beharrlichkeit, Belastbarkeit, Skalierbarkeit berücksichtig werden.

Was die Beharrlichkeitstechnologie einer Entität an geht, es müssen einige Anforderungen erfüllt werden. Erstens müsste eine effiziente Rohspeicherung von Entitäten nach Entitäten ID geben, zweitens müssen Daten konfigurierbar repliziert werden, drittens muss es Optionen für den Betrieb mehrerer Datencenter mit automatischer Replikation geben. Weiterhin eine robust hohe Verfügbarkeit, Betriebsunterstützung, Überwachung, Backup sowie Fehlerkorrektur müssen möglich sein.

Alle Dienste müssen bei Bedarf hochverfügbar sein. Alle erkannten Dienste müssen auf Integrität geprüft werden, und diese Integritätsprüfung muss sowohl intern als auch extern für Überwachungszwecke verfügbar sein[DIB18]. Ein Neustart muss durchgeführt werden für fehlgeschlagene Dienste oder die Verschiebung von neuen Instanzen auf den fehlerfreien Server muss erfolgen. Außerdem die Bereitstellung einer Programmierschnittstelle für den internen Status des Knotens durch jeden Dienst muss erfolgen. Durch einen einzelnen Bezugspunkt muss es eine strukturierte Zusammenfassung vom Status der gesamten Plattform bereitgestellt werden, mit einer optionalen Auswahl eines Themas wie Verfügbarkeit, Gesundheit und Leistungsüberwachung. Das Verhalten der Plattform sollte wie ein gigantisches Cluster sein aus der Sicht der Außenwelt.

Was die Skalierbarkeit angeht, die Bereitstellung der manuellen horizontalen Skalierbarkeit muss so einfach wie möglich erfolgen durch den Start und die automatische Entdeckung von neuen Instanzen, sowie die Einbeziehung des Routings.

### Plattformanforderungen

Bei der Plattform wird sowohl von der Architektur als auch von der Entitätsverarbeitung ausgegangen.

Die genaue Einhaltung der Prinzipien einer nativen Cloud-Architektur sollte erfolgen um es der Plattform zu ermöglichen, so viel Möglichkeiten für Architektur- und Betriebsentscheidungen wie möglich auszuwählen[DIB18]. Im Falle einer nicht gewählten Ausführungsplattform-Technologie, die sich nicht zu den Microservices eignet, sollte ein Microservice-Design von Komponenten als Norm dienen. Alle Dienste müssen eigenständig sein, bzw. zentrale Dienste, die von den anderen Diensten abhängig sind, müssen gemieden werden. Ein Dienstentdeckungsdienst muss vorgesehen werden, mit dem das automatisierte Routing ohne manuelle Konfiguration sowie die automatische Registrierung aller Plattformdienste möglich ist. Die Erkennung von neuen Dienste wie benutzerdefinierte Umwandlungen ist erforderlich. Außerdem die Übermittlung von routing- oder ausführungsrelevante Informationen über den Dienst zum Zeitpunkt der Dienstermittlung ist auch erforderlich.

Die Über die Lade-API empfangenen Entitäten müssen mindestens eine Verarbeitung erlebt haben und wenn Fehler bestehen kann die Verarbeitung bis zur maximalen Anzahl von Wiederholungen erfolgen. Entitätsverarbeitende Dienste sollen idempotent sein und sollten sowohl die Konsistenz als auch die Qualität der Ergebnisse nicht beeinflussen, im Falle von mehrfach Wiederholung einer Entitätslast. Die Konfiguration einer beliebigen Zahl von Versionen der Transformationsdienste, dessen Anwendung auf alle Entitäten erfolgt, muss möglich sein. Eine Teilreihenfolge von Umwandlungsdiensten muss an die Umwandlungskonfiguration teilnehmen.

In BigData4Biz soll eine linguistische Berechnung erfolgen. Daher werden folgend Anforderungen in Bezug auf diese genannt. Da die linguistische Berechnung auf neuen linguistischen Daten zugehen, die eindeutige IDs als eine raumhaltende Darstellung der Textdarstellung auf der gesamten Plattform brauchen, müssen diese IDs so erstellt werden, dass das Sperren gemieden wird. Angesichts des zeitaufwendigen Aspekts der linguistischen Berechnung, seine Ausführung sollte gleichzeitig für gerade verarbeitete Entitäten erfolgen. Die Konfiguration einer beliebigen Anzahl an Versionen der Ähnlichkeitsdienste, deren Anwendung auf alle Entitäten erfolgt, sollte möglich sein. Die Erkennung des Endes von gleichzeitigen Ausführung der Ähnlichkeitsdienste sollte robust sein. Für die Registrierung von Datenquellen und Agenten, liefert die Management-API einen Dienst zur Registrierung eines Agententyps, seine konfigurierte Datenquelle nach Namen und anderen Identifikationsinformationen. Die zusätzliche Identifikationsinformation muss genügend sein zur Identifizierung der tatsächlichen physikalischen Datenquelle sowie des von der Datenquelle abgedeckten Aspekts.

Außer eine Plattform BigData4Biz enthält auch Komponente, die bestimmte Anforderungen haben.

### Komponentenanforderungen

BigData4Biz enthält folgende Komponente: Agenten, der Ladedienst, Transformationsdienste, der Entitätsdienst, der linguistische Dienst, der Ähnlichkeitsdienst, der Lebenszyklusdienst, der Benachrichtigungsdienst, der Löschdienst und das Kundenprofil.

Agenten sind Microservices, die zur Ermittlung von Informationen an die Plattform dienen. Diese können konfiguriert werden für eine Datenquelle, haben die Aufgabe Daten in einem einheitlichen Format (Entität) zu bringen und an die Plattform zu senden. Darüber hinaus sollten diese Agenten autonome Dienste sein, die sich um die Überwachung und Sendung der Datenquelleninhalte an die Lade-API kümmern. Eine Interaktion mit der Plattform sollte nicht zwangsläufig sein zur Ermittlung eines Entitätsstatus oder zur Erstellung einer Entität ID. Die Verwaltung der eigenen Datenbank muss von Agenten durchgeführt werden um die Überprüfung der als Entität ID übertragenen Daten. Die Überwachung von jeder Datenquelle muss durch eine einzelne Agenteninstanz erfolgen. Es müsste eine Interaktion zwischen Agenten und Datenquelle geben, so dass die Ermittlung von neuen oder geänderten Daten möglich ist. Für neue Entitäten wird nach Möglichkeit eine Backlink-Information vergeben, derer Wahl datenspezifisch erfolgt. Weiterhin sollte es möglich sein mit dem Backlink der Ursprung von Entitäten in der physischen Quelle in Kombination mit der registrierten Datenquelle zu lokalisieren. Die Extraktion vom Textkörper einer Entität erfolgt nur dann, wenn die Daten eine Dateienart mit einem Textteil besitzen. Der eigentliche Text muss von allen technischen und strukturellen Elementen wie Sonderzeichen, Formatierung von Meta-Informationen befreit werden. Der aggregierte Text der Entität muss alle Textdaten in den Daten der Datenquelle enthalten. Der Verzicht auf Textinformationen in den Originaldaten und umgekehrt muss vermieden werden, sowie die Duplizierung von Textinformationen aus den Originaldaten im aggregierten Text. Strukturell muss eine bestimmte Reihenfolge erfolgen, nämlich erstmal den Textkörper und dann die Eigenschaftstexte. Wobei das Einfügen von Eigenschaftstexten muss als Satz erfolgen zur Erleichterung des Auftritts von Satzkooperationen. Da die Klassifizierung jeder Entität durch eine Reihe von Wörtern erfolgt, sollte die Haltung dieser Klassifizierung allgemein bleiben. Während die erste Klassifizierung die Benennung von Entität erstellende Agenten durchführt, die zweite Klassifizierung dient als „Typ“ der Entität, die immer vorhanden sein sollte. Die Kombinierung von Tabellen einer relationalen Datenquelle in einer Entität sollte möglich sein zur Vermeidung von atomare Verbindungen und Denormalisieren eines normalisierten Datenbankschemas. Die Identifizierung aller tatsächlichen Fremdschlüssel des Datensatzes muss vom Entitätsbacklink durchgeführt werden zur Ermöglichung einer Teilung von Eigenschaften einer Entität auf ursprünglichen Tabellen. Da die relationalen Datenquellen Typen über ihr Schema bereitstellen, soll die Zuordnung übereinstimmenden Entitätstyps mit dem relationalen Datentyp erfolgen. Falls der Neustart eines Agenten erforderlich ist, muss er während seinem Stillstand Informationen über Datenänderungen oder Löschungen für die entsprechenden Entitäten liefern zur erneuten Überwachung der Datenquelle. Wenn für einen Agenten die Übergabe seiner Entität an die Lade-API unmöglich ist, sollte er die Speicherung und die Wiederaufnahme der Entität durchführen, sobald das Lade-API wieder operativ ist. Im Falle einer Ablehnung von einer Entität aus Verifizierungsgründen, muss für diese Entität ein separates Protokoll durchgeführt und einen erneuten Versuch muss vermieden werden. Technische Attribute dienen zur Speicherung von nicht auf der Client-Seite gegenüberliegenden Eigenschaften. Die Verwendung von strukturellem Wissen wie Eltern-Kind-Beziehungen oder Fremdschlüsselbeziehungen ist möglich zum Einfügen von technischen Attribute. Zur Erkennung des Attributes durch den jeweiligen Ähnlichkeitsdienst müssen die Eltern/Kind-Attributnamen festgelegt werden. Es bestehen Agenten für CSV Dateien, das Dateisystem, die RDB Dateien, Web Dateien und XML Dateien. Was die CSV Datei betrifft, kommt es häufig vor, dass diese leeren Spaltenwerte haben, die nicht berücksichtigt sein können. Die Implementierung sollte flexibel genug sein, dass diese leeren Spaltenwerte in CSV-Dateien nicht berücksichtigt werden. Die Extraktion von so viele Datei-Metadaten wie möglich als Eigenschaften muss erfolgen. Die Extraktion von Metadaten und des Textkörpers aus den Dateitypen docx, xlsx, pptx, rtf, txt, html, und pdf, muss durch den Dateisystemagenten möglich sein. Es müsste eine Aufteilung des XML-Baumes unter Verwendung einfacher kundenfreundlicher Konfigurationswerte geben. Die Trennung von untergeordneten Elemente von einem Vorgängerelement muss möglich sein zur Erstellung von beiden separaten Entitäten. Für jede Unterteilung von Vorfahr und Kind muss ein global technischer Attributwert verfügbar sein, der den Ausdruck dieser aufgeteilten strukturellen Beziehung ermöglicht. Die XML-Elemente in roh Text sollten umgewandelt werden durch eine XSLT-Transformation, konfigurierbar durch einen Dateinamen entsprechenden regulären Ausdruck. Die Bereitstellung einer Standardumsetzung für nicht mit den konfigurierbaren Mustern übereinstimmende Dateien muss erfolgen. Diese Standardumwandlung müsste in der Lage sein, die Extraktion aller Elementinhalte als Nur-Text durchzuführen um deren Nutzbarkeit für die Textähnlichkeit zu ermöglichen. Zur Extraktion von bestimmten Elemente als Eigenschaften, ist es möglich, dass jede XML-Datei-XSLT-Transformation weitere Vorlagen umfassen. Diese Eigenschaften müssen als Teil der Extraktionsvorlage eingegeben werden. Eine mit einer konfigurierbaren Liste von Eigenschaftenextraktionsinformationen kombinierte generische XSLT-Transformation wird geliefert zur Bereitstellung der Extraktion von Nur-Text und Eigenschaften. Die vollständige XSLT-Transformation sollte nur in speziellen Fällen nötig sein. Die Zuordnung eines mit den Dateinamen übereinstimmenden regulären Ausdrucks für jede generische Transformation muss erfolgen.

BigData4Biz wird mithilfe des Frameworks Spring Boot implementiert. Zur Erleichterung der Implementierung von benutzerdefinierte Entitätstransformationsdienste muss die Bereitstellung eines Vorlagenprojekts mit Spring Boot erfolgen. Die Vorlage sollte eine Standardimplementierung für die Dienstintegration in die Plattform, die Standardkonfiguration und die generische Transformationslogik zur Modifikation von Einheiten, ermöglichen. Die Verwendung von entsprechenden Entwurfsmuster wie Vorlagenmuster ist erforderlich zur Vereinfachung der Fertigstellung und Anpassung von diesem Vorlagenprojekt. Die Implementierung von nur Entitäten lesende und neue synthetische Eigenschaften berechnende Standardtransformationen durch die Implementierung von nur eine Methode für die Berechnung der neuen Eigenschaften, muss möglich sein. Die Abfangung von jeder weiteren Verarbeitung ermöglichende Ausnahme muss durch einzelnen Transformationsdienste erfolgen.

Eine API wird vom Entitätsdienst geliefert für die Speicherung und den Abruf von Entitäten über ihre Entität ID. Ein optionaler partieller Abruf zur Vermeidung des Textkörperabrufs muss geliefert werden. Den Abruf oder die Speicherung des aggregierten Textes muss nicht nötig sein, weil dieser zur Verarbeitung vorgesehen sein sollte und die Beibehaltung der Zwischenergebnisse von Sprach- oder Ähnlichkeitsdiensten erforderlich ist. Die Wahl der Persistenz-Technologie ist erforderlich zur Optimierung der Rohspeicherung von Entitäten nach ID. Die Verwendung des Entitätsdienstes sollte nur durch den Abfragedienst erfolgen, um tatsächliche Entitäten bereitzustellen. Ein Zugriff auf die Entitäten wird erforderlich sein. Es müsste keine Abhängigkeit bestehen zwischen den Ähnlichkeitsalgorithmen und dem Entitätsdienst aus Skalierungsgründen. Für die Suche sollte der Indexzugriff nur vom Abfragedienst gebraucht werden. Falls kontextabhängige Informationen über Entitäten vom Ähnlichkeitsalgorithmus gebraucht werden, die lokale Beibehaltung dieser Information ist erforderlich. Die unveränderliche Behandlung von Entitäten über die Lade-API bis zur nächsten Aktualisierung ist erforderlich. Speziell muss eine separate Speicherung der Informationen über Entitäten wie Lebenszyklusstatus stattfinden. Die Speicherung der Entitäten muss so erfolgen, dass eine Optimierung des häufigsten Abrufs von Entitäten ohne den Textkörper erfolgt. Eine Entität und ihres Indexes müssen konsistent gespeichert werden.

Zur Vermeidung einer übermäßigen Belastung vom Müllsammler muss eine Zuweisung einer raumbewahrende ID an verschiedenen durch den linguistischen Dienst erzeugten Begriffen und Phrasen erfolgen. Die nicht Verwendung von Begriffe als externe Repräsentationen ist erforderlich.

Die Autonomie der Ähnlichkeitsdienste sollte so hoch wie möglich sein und die Wiederspiegelung dieser Autonomie sollte in der Beständigkeit erkennbar sein. Das Implementierungsdesign muss die Wiederspiegelung des Faktes, dass die verschiedenen Instanzen der Ähnlichkeitsdienste sowohl zur Ladezeit als auch zur Abfragezeit Konkurrent laufen, liefern. Die Ablehnung der Verarbeitung von einer Entität durch jede Ähnlichkeitsdienstversion ist möglich im Falle von ungeeignetem Ähnlichkeitsalgorithmus für diese Entität. Der Lebenszyklusstatus der Entität muss wiedergeben, dass die Verarbeitung der Entität durch die Ähnlichkeit stattgefunden hat. Die Unterscheidung dieses Status von einem Status, der den Hinweis auf einen nicht verarbeiteten Ähnlichkeitsdienst gibt, ist erforderlich. Es besteht bei der Abfrage-API die Annahme, dass die Strömung über große Listen ähnlicher Entitäten erfolgt. Die Lieferung der bestmöglichen Unterstützung durch den Ähnlichkeitsdienst ist erforderlich zur effizienten Unterstützung der Berechnung von solchen großen Listen ähnlicher Entitäten und zur Vermeidung von riesigen Ressourcenkosten. Die Berechnung von allen Ähnlichen Entitäten muss durch Algorithmen erfolgen zur Ermöglichung einer Sortierung nach Rangfolge. Die Darstellung dieser Ergebnisliste im Hinblick auf den Speicherverbrauch muss erfolgen. Alle Ähnlichkeiten sollten optional eine Zwischenspeicherung der neuesten berechneten Ähnlichkeiten, die raumlimitiert ist, fördern. Die Einstellung dieser Zwischenspeicherung sollte so erfolgen, dass es nur die Speicherung von bestimmten Ähnlichkeiten im Cache, deren Auswahl anhand der Datenquelle oder Klassifikation des Betreffs erfolgt, erfolgt. Die Entfernung von Zwischenspeicherzeilen ist erforderlich sobald das Löschen von Objekt- und Subjektentitäten erfolgt. Die Bereitstellung eines Vorlagenähnlichkeitsprojekts unter Verwendung von Spring Boot ist erforderlich zur Erleichterung der Erstellung von neuen Ähnlichkeitsdienste.

Die Beibehaltung des Lebenszyklusstatus „gelöscht“ von Entitäten muss sein zur Sicherstellung der richtigen Verwendung von Verweise auf diese Entitäts-ID. Es muss keine Abhängigkeit bestehen zwischen den Lebenszyklusdienst und irgendeine Art der Synchronisierung von Statusaktualisierungen. Die Berücksichtigung von Designs wie Ereignisbeschaffung ist umso erforderlich.

Es muss auch einen Benachrichtigungsdienst bestehen in BigData4Biz und dieser hat auch Anforderungen. Die asynchrone Bereitstellung von Statusinformationen um die Ladeverarbeitung einer Entität zu beendigen durch den Benachrichtigungsdienst ist erforderlich. Eine geeignete Benachrichtigungstechnologie für Unternehmensabläufe ist auch erforderlich. Das Abonnement des Benachrichtigungsdienstes sollte mit geringem Aufwand implementiert werden und über eine REST-API erfolgen.

### Anforderungen am Ähnlichkeitsalgorithmus

Ein Ähnlichkeitsalgorithmus wird von einem spezifischen Ähnlichkeitsdienst implementiert und wird genutzt sowohl zur effektiven Berechnung von Ähnlichkeiten als auch zur Berechnung der aktuellen Ähnlichkeit für eine abgefragte Entität. Dieser hat auch besondere Anforderungen was sein Inhalt, Struktur und Funktionsweise betrifft. Bei der Berechnung von Ähnlichkeiten und Ihre Gewichtungen ist die Beachtung eines optionalen Kundenprofils erforderlich. Eine Optimierung der Speicherung von Ähnlichkeiten muss für den Speicherplatzverbrauch und für den schnellen Abruf erfolgen. Die Vorbereitung aller Ähnlichkeitsalgorithmen muss zur Erleichterung einer begrenzte Zwischenspeicherung von neuesten berechneten Ähnlichkeiten erfolgen.

### High-Level Architektur

BigData4Biz wird hergestellt um gerade und zukünftig bei namhaften Kunden eingesetzt zu werden. Um diese Software langlebig und nachhaltig zu betreiben, ist die Herstellung einer Architektur nötig. [SHR11] behauptet, dass die Architektur eines Systems die Strukturen des Systems, dessen Bausteine, Schnittstellen und deren Zusammenspiel beschreibt. Dies Bedeutet, dass eine Architektur die Systembausteine sowie ihre Beziehungen zueinander zum einen vorstellt und zum anderen zeigt die Gruppierung von diesen Bausteinen sowie die Bausteine gehörenden Schnittstellen. Eine Architektur hilft dabei nicht nur einen Überblick über die Struktur eines Systems zu haben durch die grobe Anzeige von nur wichtige Aspekte des Systems, sondern auch zur wartbare, flexible, verständliche und langfristige Konstruktion von diesem.

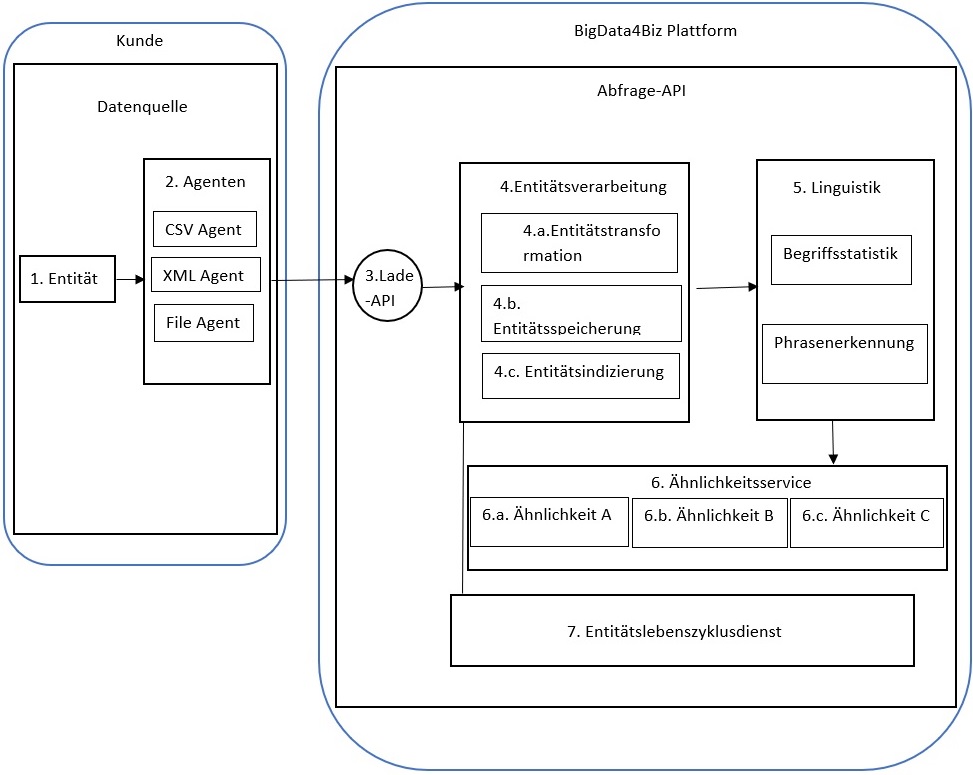


Abbildung 1:High-Level-Architektur von BigData4Biz [DIB18]

In Abb. 1 wird die Architektur von BigData4Biz gezeigt. Hier wird keine konkrete Plattformkommunikationstechnologie vorgestellt, sondern die generellen Elemente, die wichtig sind für die Funktionsweise.

Die Datenquelle ist ein wichtiger Teil der Architektur, der sich aber nicht in BigData4Biz befindet, sondern bei der Kundenseite. Diese entspricht eine physische Instanz, an der Daten generiert werden. Eine relationale Datenbank, ein Dateisystem, eine Ereignisquelle oder eine Webseite sind Beispiele von Datenquellen. Das Ziel der Datenquelle ist die Sammlung aller technischen Informationen, die einen Zugriff auf Informationen ermöglichen. Die Existenz von mehreren Datenquellen für eine physische Quelle zur Unterscheidung der logischen Partitionierung von Daten an der physischen Quelle, ist möglich. Die Datenquelle ist mit einer standardisierten Übertragungsschnittstelle versehen und enthält unterschiedliche Entitäten (1) und Agenten (2), die wichtig sind für die spätere Ausnutzung von Daten in BigData4Biz. Für die Datenquelle die Ausgabe von Entitäten verschiedener Klassifizierungen und Strukturen ist möglich.

1. Die Entität (1) stellt die primär verknüpften Daten in BigData4Biz und ist ähnlich zu einer Ressource in der Terminologie von verknüpften offenen Daten und Resource Description Framework (RDF) [DIB18]. Es besteht jedoch eine starke Unterscheidung zwischen die Datenquellen der Entitäten, ihre Verwendung und die verknüpften offenen Daten. Die Datenquelle liefert die Darstellung eines beliebigen Datenelementes aus einer Datenquelle und entspricht ein relationaler Datensatz, ein Join von Beziehungsdatensätzen, eine Datei, eine Webseite, ein Text oder allgemein strukturierte Daten aller Art. BigData4Biz kann die Berechnung der Beziehung zwischen eine Subjektentität und eine Objektentität durchführen. Es sei denn eine Verknüpfung der Subjektentität über ein Prädikat mit einer Objektentität erfolgt und die Begründung von verschiedenen Arten von Ähnlichkeiten zwischen Subjekt und Objekt ist durch das Prädikat möglich. Daten werden in einer Entität (1) durch Agenten (2) umgewandelt um später in BigData4Biz über eine Lade-API (3) geladen zu werden zur Durchführung der Dokumentähnlichkeitsbestimmung. Sobald eine Entität in BigData4Biz empfangen und gespeichert wird, besitzt diese Informationen wie die Entitätsbezeichnung, die Datenquellenbezeichnung, der Agentenname, das Backlink, Metadaten und Eigenschaften.
2. Die Übertragung der Entitäten (1) von einer Datenquelle zu BigData4Biz erfolgt in der Extract-Transform-Load (ETL)- Methode. Jedoch gibt es keine festgelegte Implementierung von diesem ETL-Schritt, wo es ausnahmsweise die Lade-API (3) gibt, deren erwarteten Aufgabe den Empfang von Entitäten (1) ist. Die Entitätsagenten (2) werden benutzt von der gebräuchlichsten Implementierung des ETL. Die Entitätsagenten entsprechen kleine Programme mit Zugriff auf eine Datenquelle, erhöhten Rechte, und die die Ermittlung von neuen oder geänderten Daten durchführen. Der Entitätsagent (2) führt die Umwandlung der extrahierten Daten in die Entitätsform selbst durch. Der Entitätsagent (2) befindet sich in der Sicherheitszone der Datenquelle zur effektiven Erkennung der Datenänderungen und einfachere Sendung von Daten. Es bestehen schon definierte Standardagenten für allgemeine physische Datenquelle, die den vollständigen ETL-Schritt liefern. Die Entitätsagenten (2) sind Buchhalter aller aus ihrer Datenquelle extrahierte Entitäten. Damit können Datenänderungen an Entitäten in BigData4Biz verfolgt werden und die Entität kann zum Zeitpunkt des Ladens aktualisiert werden.

Abfrage-API: dafür da eine Interaktion zwischen BigData4Biz und den Nutzer zu ermöglichen und entspricht einen REST-fähiger Dienst, der dazu dient verbundene Einblicke in die geladenen Entitäten abzufragen. Es können hier traditionelle Suche nach Entitäten mit Phrasen oder Textausschnitten durchgeführt werden im Rahmen von Abfragen. Die Umwandlung eines gefundenen Interessenbereichs in einen neuen Bereich, wo Ähnlichkeitsbeziehungen, zusätzliche Ausdrücke oder eine Auswahl signifikanter Ausdrücke des Geltungsbereichs benutzt werden, ist möglich. In einem Bereich befinden sich einige Sehenswürdigkeit darstellende dedizierte Entitäten und einen Kontext von zum Definieren von Ähnlichkeitsbeschränkungen verwendete Entitäten.

1. Die Lade-API (3) entspricht ein REST-konformer Dienst von BigData4Biz, die für den Empfang und die Ladung von Entitäten (1) zur weiteren Verarbeitung in BigData4Biz dient.
2. Die Entitätsverarbeitung (4) ist der Teil der Architektur, der sich um die Entitätsversorgung kümmert in BigData4Biz nachdem diese über die Lade-API (3) geladen wurde. Die Entität wird verarbeitet um passend zu werden für die verschiedenen Algorithmen, die anwesend in BigData4Biz sind. Bei der Entitätsverarbeitung erfolgen die Entitätstransformation (4.a), die Entitätsspeicherung (4.b) und die Entitätsindizierung (4.c). Die Entitätsindizierung (4.c) zum Beispiel ist ein wichtiger Prozess, der die Assoziation eines Vokabulars aus Schlüsselwörtern und allen Dokumenten eines Textkorpus durchführt.
3. Die Linguistik (5) benutzt statistische Informationen über die Texte der Entitäten erstens zur Unterscheidung der signifikante von nicht signifikanten Teilen des Textes und zweitens sowohl zur Bestimmung der Beziehungen von Text auf der lexikalischen Ebene als auch zur Strukturierung von Texten in Gruppen ähnlicher Themen. Dieses Service stellt eine gute grobe Klassifizierung bereit. Dabei wird das Service als Basis statistische Zahlen bezüglich der Häufigkeit und des Auftritts von Begriffen in Dokumente haben und nicht das Verständnis der Bedeutung von Texten. Die Linguistik befindet sich in der Abfrage-API, wo ein konkreter Benutzer Suchbegriffe eingeben kann, die später zum Informationenvergleich ausgenutzt werden. Die Verwendung von einer begrenzten Teilmenge der semantischen Analyse wie Teil der Sprachmarkierung für verbesserte Ergebnisse durch signifikante Phrasenextraktion und Begriff gemeinsames Auftreten ist möglich.
4. Die Ähnlichkeitsdienste (6) berechnen die Ähnlichkeiten zwischen den Dokumenten. Es besteht bei Entitäten eine Subjekt-Prädikat-Objekt-Beziehung (SPO-Beziehung). Die Bestimmung des Prädikates erfolgt nicht durch manuelle Zuweisung oder Berechnung unter Verwendung einer Ontologie. Ähnlichkeitsdienste (6) verfügen über Ähnlichkeitsalgorithmen, die die Berechnung von durch Prädikaten ausgedruckte Ähnlichkeiten durchführen. Die Kernähnlichkeitsalgorithmen von BigData4Biz haben als Basis linguistischen Statistiken wie TF-IDF. Die Ähnlichkeitsdienste (6) sind unabhängig von den anderen Diensten und benutzen ihre eigene Persistenz zur effektiven Berechnung von SPO-Beziehungen[DIB18].

Das Entitätslebenszyklusdienst (7) ist für die Überwachung von jeden schritten der Entitätsverarbeitung (4) zuständig.

## Wichtigen Begriffen und Definitionen aus der Ähnlichkeitsbestimmung von Dokumenten

Für die Ähnlichkeitsbestimmung von Dokumenten bestehen eine Herangehensweise und wichtigen Begriffe, die zunächst jeweils erläutert und definiert werden.

### Herangehensweise bei der Untersuchung von Dokumentähnlichkeit

Für die Untersuchung der Dokumentähnlichkeit erfolgen die linguistische Verarbeitung, die Termgewichtung, die Darstellung der extrahierten Dokumentterme als Vektoren der gewichteten Terme, die Berechnung der Ähnlichkeitswerte anhand von Ähnlichkeitsmaßen, Verwendung der ermittelnden Ähnlichkeitsmatrix für die Ähnlichkeitsbestimmung, statt [GIN13].

**Linguistische Verarbeitung:** Während diese erfolgen die Termextraktion und die Analyse von Dokumenten. Die Analyse der Dokumente erfolgt in drei Schritte, nämlich die morphologische Analyse, die syntaktische Analyse und die semantische Analyse [SAL87]. Die morphologische Analyse untersucht die Wortformen und Wortbestandteile, die Sinn tragen. Durch einen Deskriptor wird die Stammform eines Wortes festgelegt um bei Erscheinung dieses Wortes mit dem gleichen Stamm und nicht komplett die gleiche Form, das Wort erkannt wird; Die syntaktische Analyse behandelt Sätze und ihre Komponenten, d. h die Normalformen für Sätze werden definiert, um wörtergruppen zu erkennen und Deskriptoren können aus zusammengelegene Wörter einzelne Wörter erkennen. Es wird für die syntaktische Analyse eine kontextfreie Phrasenstrukturgrammatik verwendet, die einen Strukturbaum erstellt, der die Beschreibung der syntaktischen Struktur eines Satzes liefert [WUA97].

Bei der semantischen Analyse wird das komplette Text eines Dokumentes behandelt, durch Erkennung von Zusammenhänge im Dokument, die Sinn tragen. Hier wird der Kontext berücksichtigt zur Zerlegung eines Textes in Einheiten, die bedeutungsabhängig sind.

Der nächste Schritt bei der Dokumentähnlichkeitsbestimmung nach der linguistischen Verarbeitung ist die Termgewichtung.

**Termgewichtung:** Diese besteht aus drei Komponente, nämlich die Termfrequenz, die Kollektion Frequenz, und die Normalisierung der Dokumentenlänge [SAL88]. Ihre Rolle besteht darin die Informationsrückgewinnungseffektivität zu verbessern und somit wird die unter Berücksichtigung von zwei Maße, nämlich Recall und Precision ausgedruckt durch die Formel: tf \* idf

tf entspricht der Faktor Termfrequenz, die die Frequenz der Term Auftritte in einem Dokument oder Abfragetext misst. Idf entspricht der Faktor des Inverse Document Frequency, der dafür sorgt die Begriffe, die in wenigen Dokumenten einer Sammlung konzentriert sind, zu begünstigen[SAL88]. Für eine Anzahl n von Dokument und eine Sammlung N von Dokumenten er ist gegeben durch log N/n [SPA72].

Bei Systeme mit variierten Vektorlängen werden kleine Dokumente mit kleinen-Term Vektoren abgebildet und lange Dokumenten werden durch lange-Term Mengen abgebildet. Diese Situation führt zu einer Chancenungleichheit zwischen Große Dokument, mit mehr Wörter und kleine Dokument mit wenige Wörter, Wo Große Dokumente eine Bessere Wahrscheinlichkeit haben abgerufen zu werden auf Basis der Wörteranzahl. Laut [SAL88] ein Normalisierungsfaktor stellt sich als nützlich für den Ausgleich von Dokumentenvektoren und stellt sich damit als Ergänzung der Termgewichtungsformel. Darüber hinaus für das Gewicht W eines Terms t die komplette Termgewichtungsformel ergibt sich durch[SAL88]:

**Darstellung der extrahierten Dokumentterme als Vektoren der gewichteten Terme:** all die Terme, die linguistisch verarbeitet wurden, werden gefasst in einem Term Wörterbuch. Eine Menge von Dokumenten wird dann in einem Vektor dargestellt. Diese Darstellung erfolgt indem ein Dokument auf eine Kette von gewichteten Terme abgebildet wird. Diese Vektoren können binär oder reelle Zahlen zwischen 0 und 1 sein [ZOU10]. Wobei in einem Dokument binäre Vektoren weisen auf das Vorkommen eines Termen zu, während die reellen Zahlen auf die relative Häufigkeit eines Terms zu weisen. Nachdem diese Vektordarstellung stattgefunden hat, werden ähnlichkeitswerte berechnet.

**Berechnung der Ähnlichkeitswerte anhand von Ähnlichkeitsmaßen:** die Berechnung der Ähnlichkeitswerte führt zu der Erstellung einer Ähnlichkeitsmatrix, die die Ähnlichkeiten zwischen den Dokumenten-Vektoren darstellt. Es bestehen unterschiedliche Ähnlichkeitsmaßen für die Ähnlichkeitsmessung. Unter anderem können die Kosinus-Ähnlichkeit und die Jaccard-Ähnlichkeit genannt werden.

### Definition von wichtigen Begriffe aus der Ähnlichkeitsbestimmung von Dokumenten

Es werden viele fachliche Begriffe in dieser Arbeit verwendet. Für ein besseres Verständnis werden alle wichtigen Begriffe folgend definiert:

**Ähnlichkeitsmatrix:** Tabelle von Dokumenten-Vektoren, die die Ähnlichkeit zwischen diesen Dokumenten-Vektoren einer Dokumenten Sammlung zeigt mit Werten zwischen 0 und 1.

**Ähnlichkeitsmaßen:** ist eine Funktion zur Ähnlichkeitsbestimmung, die je zwei zu vergleichende Objekten eines n-dimensionalen Raum einen nicht negativer Wert zuordnet, welcher als Abstand den beiden zu vergleichenden Objekten voneinander aufgefasst wird (Anlehnung an [CAR92]).

**Normalisierung der Dokumentenlänge:**  entspricht die Reduzierung der Dokumentlänge durch Entfernung von zum Beispiel Stoppwörter, um die Dokumentenvektoren auszugleichen

**Termfrequenz:** entspricht wie häufig ein Begriff oder Term in einem Dokument vorkommt.

**Termgewichtung:** ist die Häufigkeit eines Terms in einem Dokument oder in einer Sammlung

**N-Gramm**: entspricht eine Sequenz, die geordnet ist, von n in einem Textkörper extrahierte Wörter.

**Jaccard Ähnlichkeit**: es ist ein Maß für die Ähnlichkeit für zwei Datenmengen in einem Bereich zwischen 0 und 1. Der Koeffizient der Jaccard Ähnlichkeit für zwei Dokumente a und b wird gegeben durch die Formel: Länge (schnitt (a, b)) / Länge (Vereinigung (a, b)) [MUL16].

**Match**: unter dem Begriff Match wird Übereinstimmung verstanden.

**Threshold**: entspricht einen Schwellenwert

**Hashing**: entspricht die Umwandlung eines beliebig großen Datensatz in eine Zeichenkette mit einer festen, kürzeren Länge, die den ursprünglichen Datensatz referenziert [TEC18].

**Hash-Signatur**: entspricht der Schlüssel eines Hashwerts

**Token**: entspricht eine Folge alphabetischer und/oder numerischer Zeichen, mit Trennung links und rechts durch Textformatierung und/oder Interpunktionszeichen [Yatsko2014].

**Minhash:** entspricht das gehackte minimale Index der ersten Zeile einer Tabelle für das Vorkommen eines Wortes in einem Dokument nach der Durchführung von Zeilenpermutationen.

**Begriffsfrequenz:** entspricht bei der Dokumentähnlichkeit wie häufig ein Begriff in einem Dokument vorkommt

**Begriffsbedeutung:** entspricht der Sinn, der in einem Begriff liegt

**minhash-Signaturen:** Schlüssel eines minhash-werts

**Bucket:** entspricht ein Cache für Daten.

**Recall** ist der Anteil der abgerufenen relevanten Artikel, gemessen am Verhältnis der Anzahl der relevanten abgerufenen Artikel zur Gesamtzahl der relevanten Artikel in der Sammlung [SAL88].

**Precision** ist der Anteil der abgerufenen relevanten Artikel, gemessen durch das Verhältnis der Anzahl der relevanten abgerufenen Artikel zur Gesamtzahl der abgerufenen Artikel [SAL88].

**Kandidatenpaaren:** Dokumentenpaar, die für die Ähnlichkeitsuntersuchung in Frage kommen bei den BayesLSH und BayesLSH-Lite Algorithmen.

**Dokumentähnlichkeit:** ist der Fakt, dass zwei Dokumenten gleich sind oder den gleichen Inhalt haben.

**Term:** entspricht in der Linguistik ein Begriff oder Wort.

## Ansätze zur Dokumentähnlichkeitsbestimmung

### Einführung von BayesLSH und BayesLSH-Lite

### Naive Bayes

### Pairwise Compare

Pairwise Compare ist ein Algorithmus, der die Ähnlichkeit paarweise zwischen Dokumenten schätzt. Es geht bei diesem Algorithmus, Dokumenten in Paare zu stellen, dann die Ähnlichkeit zwischen denen zu berechnen und die Als Matrix zurückzugeben.

Ein Dokument d, mit t Terme verfügt über eine Abbildung als Vektor mit Termgewichtungen für jeden Term. Der Symmetrische Ähnlichkeitsmaß ist gegeben durch [ELO08]:

(2.5.3.1)

entspricht die Ähnlichkeit zwischen den Dokumenten . V entsprich die Vokabular Menge. Die Bedingung zur Berücksichtigung des Terms, dass , großer null sind. t ist dann Element von V und kann ersetzt werden durch t ∈ di ∩ dj in der Gleichung 2.5.3.1 [ELO08]. Die Abbildung 2 zeigt den Pairwise Compare Algorithmus zur Dokumentähnlichkeitsbestimmung.

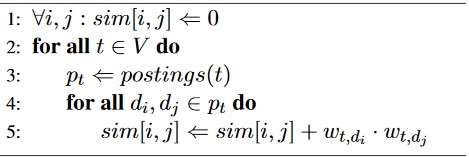


Abbildung 2: Der Pairwise Compare Algorithmus zur Dokumentähnlichkeitsbestimmung [ELO08]

Postings (t) entspricht eine Liste von Dokumenten, die einen bestimmten Term t enthalten. In diesem Algorithmus wird angenommen das die Speicherung der Termgewichtungen in Postings erfolgt.

## Vergleich von BayesLSH und Pairwise Compare

## Die Programmiersprache R

R ist eine Programmiersprache, die dazu dient statistische Berechnungen durchzuführen und Graphiken darzustellen. R wird durch die freie Software „R“ betrieben und ist ähnlich zu der S Programmiersprache. Unter „freie Software“ wird eine Nutzerfreiheit und -gemeinschaft respektierende Software verstanden. Die R Software ist kostenlos im Internet verfügbar und kann sowohl auf UNIX Plattformen, als auch auf Windows und MacOS Betriebssysteme betrieben werden. R ermöglicht es sowohl statistische Methoden wie lineare und nicht lineare Modellierung, klassische statistische Testen, Time-series Analyse Klassifikation oder noch Clustering, als auch graphische Methoden durchzuführen.

Mit R sind viele Möglichkeiten für den Programmierer verfügbar. Daten können unter anderem effektiv verarbeitet werden und der Speicher eingerichtet werden. Es bestehen Operatoren mit denen Arrays (Matrizen) berechnet werden können sowie eine Sammlung von Zwischenwerkzeugen und grafische Einrichtungen, die sich für die Datenanalyse eignen. Wie die meisten Programmiersprachen ist die Anwendung mit Bedingungen, Schleifen, benutzerdefinierte rekursive Funktionen, sowie Eingabe- und Ausgabefunktionen versehen. R ist flexibel und erweiterbar, d. h Sie verfügt außerdem auch über Pakete, die sowohl durch der R Distribution als auch durch die CRAN-Familie im Internet erhältlich sind.

# Verwandte Arbeiten

# Spezifikation und Design

In diesem Kapitel werden Spezifikation und Design des BayesLSH-Lite Algorithmus erläutert. Die Spezifikation dient dabei Anforderungen an dem BayesLSH-Lite Algorithmus zu geben und das Design zeigt wie der Entwurf des Algorithmus aussieht.

## Spezifikation

Eine Spezifikation eines Algorithmus besteht aus einer präzisen Beschreibung des Verhaltens des Algorithmus und einer präzisen Beschreibung der geforderten Eigenschaften an die Implementierung des Algorithmus (Anlehnung an [HAJ10]).

Die Spezifikation wird eine Beschreibung von den angeforderten Fähigkeiten eines Systems unter gegebenen Bedingungen gegeben. Eine Spezifikation kann in vier Teile gegliedert werden: die Beschreibung der Aufgabenstellung, die Beschreibung der Schnittstelle und die Beschreibung von Funktionalen und nicht funktionalen Anforderungen.

### Aufgabenstellung

Die Aufgabenstellung kann durch eine Funktions- oder ein Zustandsschema beschrieben werden. Bei dem Funktionsschema wird eine Funktion, die aus Eingangsgrößen Ausganggrößen zurückgibt durch mathematischer Formel definiert und bei einem Zustandsschema erfolgt die Beschreibung des Anfangszustands und des Endzustands, der als Folge der Ausführung eines Algorithmus dient [POH08]. Für die Aufgabenstellung habe ich mich für das Zustandsschema entschieden:

**Aufgabe:** Bestimmung der Ähnlichkeit zwischen Dokumenten

**Gegeben:** eine Liste von Dokumentpaaren C, Ein Ähnlichkeitsgrenzwert t, ein Abrufsparameter a, Anzahl der zu verwendenden Hashes h

**Gesucht:** Eine Menge O von Dokumentenpaaren (x, y) zusammen mit genauen Ähnlichkeiten Ŝ x, y

Nach der Beschreibung der Aufgabenstellung kann weiterhin die Beschreibung der Schnittstelle erfolgen.

### Beschreibung der Schnittstelle

Die Schnittstelle kann durch einen Pseudocode mit genauer und detaillierter Beschreibung der jeweiligen Parameter erläutert werden.

**Pseudocode: Algorithmus BayesLSH-Lite**

**Entitäten beschneiden:** Abbau der falsch positiven Entitäten paare /Dokumentenpaare

H(x): Hash des Dokuments x aus dem Dokumentenpaar (x, y)

H(y): Hash des Dokuments y aus dem Dokumentenpaar (x, y)

Pr: = Wahrscheinlichkeit, dass die Ähnlichkeit ist großer als der Ähnlichkeitsgrenzwert t  
S: = Ähnlichkeit (zufällige Variable bei Pr)  
p: = posterior Verteilung (Bayes Satz)

C: =Array von I Entitätsobjektpaare/Dokumentenpaare (Menge von Dokumentpaare, die vorher in BD4B auf Ähnlichkeit geprüft waren)

I: = Array von zwei Dokumente (Dokumentenpaar)

: genaue Ähnlichkeit zwischen Dokumente x und y

t: = Ähnlichkeitsgrenzwert für die zu vergleichenden Dokumentenpaare (wird vom Benutzer spezifiziert)

a: = Abrufsparameter

h: = Anzahl an Hashes, die zu benutzen sind

int n: = Anzahl an Hashes

int m: = Übereinstimmung von Dokumenten

M (m, n): = Ereignis von m Übereinstimmungen aus den n ersten Hashes

Boolean istbeschnitten: Variable zum Prüfen ob ein Entitätspaar/Dokumentenpaar beschnitten wurde oder nicht

String x, y = Entitäten/Dokumenten von Array I

**for all** I [x, y] ∈ C **do**

n = 0

m = 0

istbeschnitten ==false

**While** n < h **do**

m = m + Summe von n zu n + k von Hi(x). compareto(Hi(y)) (Summe von allen übereinstimmenden Hashes für ein Dokumentenpaar)

n = n + k

**if** Pr[S>=t | M (m, n)] < a **then**

istbeschnitten == true

**break** (beschneide die Dokumentenpaare)

**end if**

**end while**

**if** istbeschnitten == false **then**

= aehnlichkeit (x, y) (genaue Ähnlichkeit)

**if**  > t **then**

O. add (I (x, y), )

**end if**

**end for**

**Return** O

Es gilt zunächst die funktionalen Anforderungen an das BayesLSH-Lite Algorithmus

### Funktionale Anforderungen an das BayesLSH-Lite Algorithmus

Hier sollen einige Fragen beantwortet werden, nämlich was soll der Algorithmus leisten? Welche Dienste soll er anbieten? Was sind die Eingaben, Verarbeitung und Ausgaben von diesem Algorithmus? Wie soll er sich in bestimmten Situationen verhalten? Die Antwort zu diesen Fragen folgt dann zunächst.

Allgemein muss der Algorithmus in der Lage sein, Ähnlichkeiten zwischen Dokumenten zu berechnen. Dafür müssen Elemente wie die Datenquelle, die Datenmenge, die Datenqualität, das Dokumentformat, der Nutzer, die Plattform BigData4Biz berücksichtigt werden.

Bezüglich der Datenquelle, der Algorithmus muss in der Lage sein, Dokumente von einem gegebenen Speicherort abzurufen. Dieser sollte auch in der Lage sein auf dem Dokumentinhalt zuzugreifen, bzw. der sollte in der Lage sein das Dokumentformat zu lesen. Es bestehen dann unterschiedliche Dokumentformate, nämlich pdf, CSV, JSON, XML, TXT, JS (JavaScript), docx, xlsx, pptx, rtf, html, die er behandeln muss. Der zu untersuchende Speicherort muss vom Nutzer eingegeben werden. Falls er mehrmals ausgeführt wird mit Verweis auf den gleichen Speicherort, muss der in der Lage sein immer neu ausführbar zu sein und die neuen oder geänderten Dokumente in dem Speicherort zu erkennen und aufzunehmen. Falls der Speicherort keine Dokumente enthält sollte der Algorithmus eine Fehlermeldung anzeigen mit Benennung des Problems. Im Falle von fehlerhafte Dokumente in einem Speicherort, sollte der Algorithmus in der Lage sein eine Fehlermeldung diesbezüglich und der Name des fehlerhaften Dokuments anzuzeigen. Der Algorithmus sollte in der Lage sein, die Anzahl an Dokumente bestehend in einem Speicherort anzugeben.

Außerdem für den Fall der Dokumentähnlichkeit wird schon eine Menge von Dokumenten benötigt. Der Algorithmus sollte schon skalierbar sein, d. h der sollte in der Lage sein nicht nur eine kleine Menge an Dokumenten zu vergleichen, sondern auch große bis sehr großen Dokumentenmengen zu vergleichen, da BigData4Biz auch für Kunden mit umfangreichen Datenspeicher gedacht wird. Bezüglich der Dokumentgroße, er sollte in der Lage sein sowohl kleine Dokumente als auch große Dokumente zu vergleichen.

Der Algorithmus sollte auch in der Lage sein die genaue Ähnlichkeit zwischen den Dokumenten zu berechnen und entweder eine Ähnlichkeitsmatrix oder eine Liste der Dokumente mit den entsprechenden Ähnlichkeiten zurückzugeben. Es sollte möglich sein, Übereinstimmungen mit einem bestimmten Dokument aufgelistet anzuzeigen durch diesen Algorithmus.

Für den Abschluss des Teils Spezifikation werden nicht funktionale Anforderungen an das BayesLSH-Lite Algorithmus gegeben.

### Nicht funktionale Anforderungen an das BayesLSH-Lite Algorithmus

Unter nicht funktionale Anforderungen an das BayesLSH-Lite Algorithmus können technische Anforderungen, Qualitätsanforderungen und Anforderungen an die Benutzbarkeit des Algorithmus genannt werden.

**Technische Anforderungen:** Der Algorithmus muss in R geschrieben werden und muss in der BigData4Biz Plattform laufen. Das bedeutet, dass ein Nutzer sollte schon in der Lage sein durch Betätigung einer Funktion auf die Plattform, den Algorithmus aufzurufen oder auszuführen. Falls der Nutzer ein Bereich auswählt auf die Plattform, sollte es möglich sein diesen Bereich als Speicherort in dem Algorithmus automatisch einzutragen. Der Algorithmus soll so geschrieben werden, dass die Ähnliche Dokumenten leicht zu finden sind für Leute ohne IT Hintergrund. Es müsste möglich sein zu signalisieren, wenn neue Dokumente geladen werden in einem Speicherort.

**Qualitätsanforderungen** (Performanz und Zuverlässigkeit): der Algorithmus muss bei 99 % verfügbar sein. Die Datenmengen, die in Frage kommen für den Vergleich können hinsichtlich des verfügbaren Speichers sehr groß sein und die Ähnlichkeitsbestimmung könnte daher aufwendig sein. Trotz dieser Situation, sollte der Algorithmus innerhalb von höchstens 90 Sekunden Ergebnisse liefern. Die gelieferten Ergebnisse müssen so genau wie möglich sein.

**Anforderungen an die Benutzbarkeit des Algorithmus:** Es sollte für die Nutzer möglich sein, über die Plattform von BigData4Biz die Dokumente mit der höchsten Übereinstimmung anzuzeigen, nachdem der Algorithmus den Dokumenten verglichen hat.

## Design

In diesem Unterkapitel wird das Design des BayesLSH-Lite Algorithmus vorgestellt mit Berücksichtigung der vorher beschriebenen Spezifikation. Das Entwurfsmuster mit der Schablonenmethode (oder Vorlage Methode) in Unified Modeling Language (UML) wird benutzt um das Algorithmus Design darzustellen. Der Zweck dieser Methode ist die Definition des Skelettes eines Algorithmus in einer Operation und das Delegieren einzelnen Schritten an Unterklassen [GHR96].

Das BayesLSH-Lite Algorithmus entwickelt von Satuluri und Parthasarathy in [SAP12], gibt Auskünfte über wie Anhand von der LSH und Minhash Methoden sowie die Posterior Verteilung, Ähnlichkeiten zwischen Dokumenten berechnet werden können.

Die Abbildung 3 zeigt zwei Hauptklassen, nämlich die Klasse Dokument und die Klasse LSH. Diese beiden Klassen sind „Eltern Klassen“ für jeweils die Unterklasse Entität und die Unterklasse BayesLSH-Lite. Die Klasse Dokument definiert die Klasse Entität, während der Klasse LSH die Klasse BayesLSH-Lite definiert. Die Klasse Dokument kann unter anderem die Funktionen oeffne (), lese () und schliesse () ausführen. Die Klasse Entität ist eine Spezialisierung der Klasse Dokument und erbt die Funktionen oeffne () und lese ().

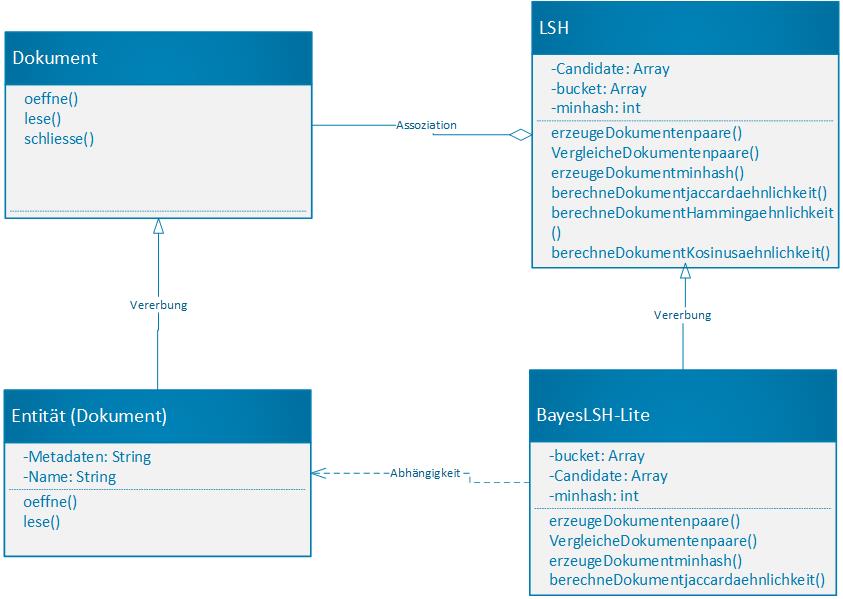


Abbildung 3: Design BayesLSH-Lite Algorithmus

Es besteht eine Assoziation zwischen die Klassen Dokument und LSH, da sich eine Vererbung in diesem Fall nicht eignet. Die Klasse LSH kann Funktionen in Bezug auf die Klasse Dokument definieren.

Die Klasse LSH besteht aus drei Attributen, nämlich Candidate (vom Typ Array), bucket (auch vom Typ Array), minhash (vom Typ integer) und sechs unterschiedlichen Funktionen, nämlich

erzeugeDokumentenpaare (),

VergleicheDokumentenpaare (),

erzeugeDokumentminhash (),

berechneDokumentjaccardaehnlichkeit (),

berechneDokumentHammingaehnlichkeit (),

und berechneDokumentKosinusaehnlichkeit (),

von denen die Klasse BayesLSH-Lite jeweils allen Attributen und vier Funktionen erbt. Die Abhängigkeit zwischen den Klassen Entität und BayesLSH-Lite ermöglicht es der Klasse BayesLSH-Lite geerbte Funktionen in Bezug auf die Klasse Entität auszuführen und somit nicht nur allgemein für Dokumenten, sondern spezifisch zu den bestehenden Entitäten für unser Anwendungsszenario. Es können unter anderem auf Entitäten Basis minhashes generiert werden, Dokumente paare verglichen werden, und die spezifische Jaccard Ähnlichkeit für Dokumenten Paare berechnet werden.

# Implementierung

Dieses Kapitel befasst sich mit der Implementierung des BayesLSH-Lite und des Pairwise Compare Algorithmus zur Dokumentähnlichkeitsbestimmung. Es werden unter anderem die ausgewählte Programmiersprache und die wichtigen Pakete sowie Algorithmen dieser Implementierung erläutert.

## Die ausgewählte Programmiersprache: R

Die Firma dibuco GmbH entwickelt normalerweise Softwarelösungen in den Programmiersprachen Java oder Scala aber die verwendete Programmiersprache für diese Thesis ist R, und die integrated development environment (IDE) dazu ist Rstudio. Ich habe mich für die Programmiersprache R entschieden, da R Statistik mit Datenvisualisierung kombiniert und dies optimal zum zentralen Thema dieser Arbeit Bayessche Statistik, passt. Dies hat unter anderem unterschiedliche Gründe. Es bestehen schon in R Bibliotheken und vorgefertigte Algorithmen, die zur Anwendung bereitsteht. R kann unter anderem sowohl mit Spark (Big Data Framework für Analytik) als auch mit Konstanz Information Miner Analytik Plattform (KNIME) betrieben werden und fast alle Datenbanken können von der ausgelesen werden. Das Textreuse Paket, das später benutzt wird für die Implementierung des Ansatzes kommt vom Massachusetts Institute of Technology (MIT). Hinter R steht eine sehr große, weltweite Community, welche die Pakete frei zur Verfügung stellt, weiterentwickelt und pflegt. Ein weiterer Vorteil von R wie schon im Abschnitt 2.7 erwähnt, ist, dass, sie eine Open Source ist. Außerdem viele wissenschaftliche Arbeiten können in R erstellt werden.

## Wichtige Pakete

Wichtige Pakete für die Implementierung dieser Arbeit sind die Pakete textreuse für die Dokumentähnlichkeitsbestimmung und die Pakete für den Zugriff und das lesen der unterschiedlichen Dateiformate, dessen Beschreibungen folgend gegeben werden.

**Das Paket textreuse:** Das Paket textreuse ist verfügbar auf die R Plattform und enthält eine Menge von Funktionen, zur Ähnlichkeitsmessung zwischen Dokumenten sowie Erkennung von wiederverwendeten Textpassage in Dokumente. Das Paket textreuse verfügt über zwei Klassen und implementiert unterschiedliche Funktionen. Im textreuse Paket sind die Klassen TextReuseTextDocument und TextReuseCorpus bestehend. In die erste genannte Klasse sind der Text eines Dokuments und seine Metadaten enthalten. Durch Eingabe der entsprechende Speicherort und Dokumentenname wird ein Dokument geladen und sein Text automatisch in Token übersetzt. Die Klasse ermöglicht es dem Programmierer zu entscheiden, ob die Tokens beibehalten werden oder nicht. Eine weitere Aufgabe der TextReuseTextDocument Klasse besteht darin die Tokens in Hashes zu transformieren anhand einer Hashfunktion. Durch zwei unterschiedliche Funktionen kann auf die Hashes zugegriffen werden. Zum einen mit der Funktion hashes () kann einen Zugriff auf die komplette Anzahl an hashes, die früher Token waren und nun eine integer Darstellung von diesen liefern und zum anderen mit der Funktion minhashes (), die eine Signatur liefert, als Repräsentation eines kompletten Dokuments. In der zweiten Klasse des Paketes textreuse, nämlich TextReuseCorpus enthält eine Sammlung von Dokumenten und wird gegeben durch den Verweis auf einen entsprechenden Ordner. Wie die Klasse TextReuseTextDocument kann diese Klasse den Text der jeweiligen Dokumente in Token übersetzen. Die von diesem Paket implementierte Funktionen sind minhash, Locality Sensitive Hashing sowie Ähnlichkeitsfunktionen.

**Die Pakete für den Zugriff und das lesen der unterschiedlichen Dateiformate**: Aus den Anforderungen an BigData4Biz und der Spezifikation des Algorithmus BayesLSH, ist Voraussetzung, dass das BayesLSH-Lite Algorithmus unterschiedliche Datei-Formate lesen kann. In R ist es schon möglich einige Dateien auszulesen. Jedoch wird bei bestimmte Dateien Typ unterstützende Pakete gebraucht. Darüber hinaus einige Pakete zum Auslesen von bestimmte Datei-formate stehen zur Verfügung. Unter anderem können die folgende Pakete genannt werden: rjson, pdftools, csv, readtext, readxl, xml2. Entsprechend werden jeweils diese o. g Pakete zum Auslesen von Json-Dateien, pdf-Dateien, csv-Dateien, txt-Dateien, xls-Dateien und XML-Dateien genutzt.

## Der Algorithmus BayesLSH-Lite

Der für diesen Implementierungsteil angewendete Algorithmus ist der BayesLSH-Lite Algorithmus. Der BayesLSH-Lite Pseudocode wurde beschrieben im Kapitel 2. Es ist hinzuweisen, dass diesen Pseudocode als Leitfaden für die Implementierung des Ansatzes genommen wurde, bzw. für das Schreiben des Codes. Weitere Details werden im Abschnitt 5.3.1 erläutert.

### Der Code

Der Code dieser Implementierung wurde in R geschrieben und anhand vom existierenden Paket textreuse durchgeführt. Zusätzlich zu dem Paket textreuse wurde auch das Paket rjson zum Auslesen von JSON Dokumente in diesem Code angewendet wie im Abbildung 2 gezeigt wird.

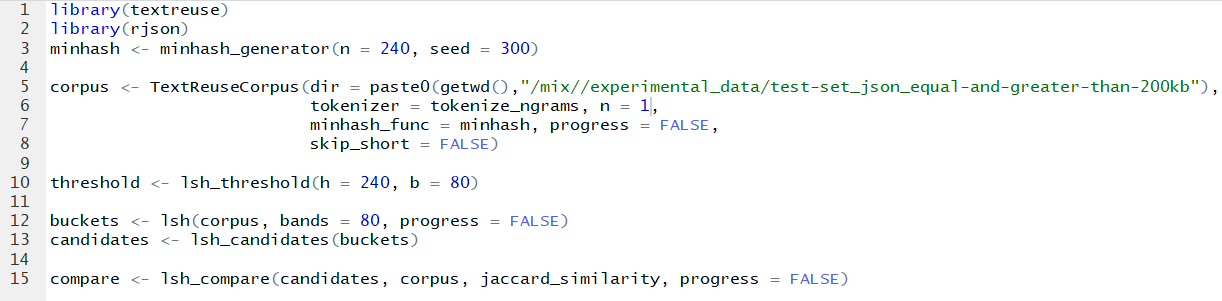


Abbildung 4: Der Code zur Implementierung des BayesLSH-Lite mit dem Jaccard Index

**Die Funktion minhash** (Noch zu definieren in Grundlagen Abschnitt mit BayesLSH): diese implementiert die Funktion minhash\_generator () zur Umwandlung von einem Text, der in Token übersetzt wurde, in Hash. Hier ist hinzuweisen, dass die Tokens zufällig ausgewählt werden. Durch Hashing von String in Zeichenvektor auf ganze Zahlen und Auswahl der Minimalwert erfolgt die Berechnung eines minhashes. Unter Hashing wird die Umwandlung eines beliebig großen Datensatz in eine Zeichenkette mit einer festen, kürzeren Länge, die den ursprünglichen Datensatz referenziert, verstanden [TEC18]. Die Definition von n Anzahl an Hashes, die von zufällige selektierte Token zu gehackte Token werden, ermöglicht es der Funktion bis zu n Hashes zu produzieren. Durch den Parameter seed kann die mehrmalige Generierung von minhashes durch eine gleichen minhash Funktion sichergestellt werden. Diese Funktion wird dann später als Parameter hash\_func in der Funktion TextReuseCorpus () angewendet.

**Die Funktion Corpus:** diese Funktion implementiert die Funktion TextReuseCorpus () und enthält eine Liste von Objekten der Klasse TextReuseDocument (). Die Klasse TextReuseDocument () enthält einen Dokumenttext und ihre Metadaten. Die Funktion Corpus ist anhand eines Dateienordners erstellt. Hier in diesem Code durch den Parameter *dir* wird die Zuweisung zu einem Dateienordner gemacht. Diese Funktion verhält sich so, dass sobald ein Dokument oder eine Datei geladen wird, der Text davon wird in Token umgewandelt. Das Ganze wird durch den Parameter tokenizer eingestellt für n-gramm. Weiterhin wenn es Dokumente bestehen, die klein sind, wird die Funktion Corpus diese einfach trotzdem berücksichtigen durch den Parameter skip\_short.

**Die Funktion threshold:** implementiert die Funktion lsh\_threshold, die dabei hilft bei der Bestimmung der minimalen Jaccard Ähnlichkeit um zwei Dokument als matches anzunehmen. Das Threshold ist durch die Folgende Formel gegeben [Leskovec2014]:

r ist die Anzahl an Spalten und wird durch die Formeln

gegeben. Wobei h und b jeweils die Anzahl an minhash-Signaturen und die Anzahl an LSH-Bänder entsprechen.

**Die Funktion buckets:** implementiert die Funktion lsh (), deren Aufgabe die Entdeckung von möglichen Matches eines gegeben corpus von Dokumenten (Sammlung von Dokumenten von der Funktion TextReuseCorpus) zum schnellen Vergleich von nur ähnliche Dokumentenpaaren dieser Sammlung, ist. Dafür werden jede Hash-Signaturen in Bänder geteilt, die eine bestimmte Anzahl an Zeile haben und es erfolgt das Hashing von diesen Bändern. Die Dokumente werden so aussortiert, dass wenn die gleichen Zeilen haben, dann wird das Hashing von beiden in den gleichen Bucket durchgeführt. Buckets können als Cache bezeichnet werden. Diese Funktion ist versehen um später in der Funktion lsh\_candidates () angewendet zu werden um potentialen Kandidatenpaare zu berechnen.

**Die Funktion Candidates:** implementiert die Funktion lsh\_candidates () und aus den Buckets der Funktion LSH () mögliche Kandidatenpaaren zurückgibt. Diese Funktion wird in der

**Die Funktion compare:** implementiert die Funktion lsh\_compare (), deren Aufgabe die Kandidatenpaare zu Vergleichen und die Entsprechende Jaccard Ähnlichkeit zu bestimmen und auszugeben.

## Der Pairwise Compare Algorithmus zur Dokumentähnlichkeitsbestimmung

Der Pairwise Compare Algorithmus für die Dokumentähnlichkeitsbestimmung ist ein Algorithmus, der aus eine Liste von Dokumente, Dokumenten in Paare stellt und vergleich mithilfe eines Ähnlichkeitskoeffizients.

### Der Code

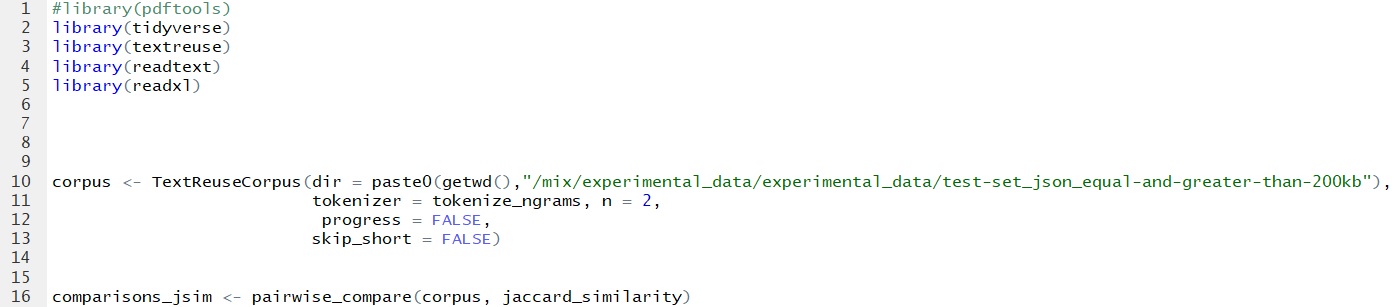


Abbildung 5: Der Code für das Pairwise Comparision Algorithmus

**Der Parameter Corpus:** diese implementiert die Funktion TextreuseCorpus (), die dabei eine Liste von Dokumenten aus einem Speicherort. Die Funktion Corpus ist anhand eines Dateienordners erstellt. Hier in diesem Code durch den Parameter *dir* wird die Zuweisung zu einem Dateienordner gemacht. Wenn es Dokumente bestehen, die klein sind, wird die Funktion Corpus diese einfach trotzdem berücksichtigen durch den Wert FALSE für den Parameter skip\_short.

**Der Parameter comparison\_jsim:** Der implementiert die Funktion paiwise\_compare (), die Dokumente in Paare stellt und ihre entsprechenden Ähnlichkeit berechnet. Die Ausgabe dieser Funktion ist eine Matrix mit Dimensionen, die die Länge des Parameters Corpus entspricht und Zeilen sowie Spalten gezeichnet mit den Dokumentennamen. Durch die Funktion Jaccard\_Similarity () wird die entsprechende Ähnlichkeit zu einem Paar des Corpus in der Matrix berechnet.

# Evaluation

In diesem Kapitel wird erläutert welche Experimente durchgeführt wurden, welches Setup, welche Datenbasis, grob der Versuchsaufbau für jeden Algorithmus. Die Ergebnisse werden präsentiert, im Anschluss werden diese Ergebnisse diskutiert und Schlussfolgerungen gezogen. Allgemein wird auch die Skalierbarkeit von den jeweiligen Algorithmen evaluiert.

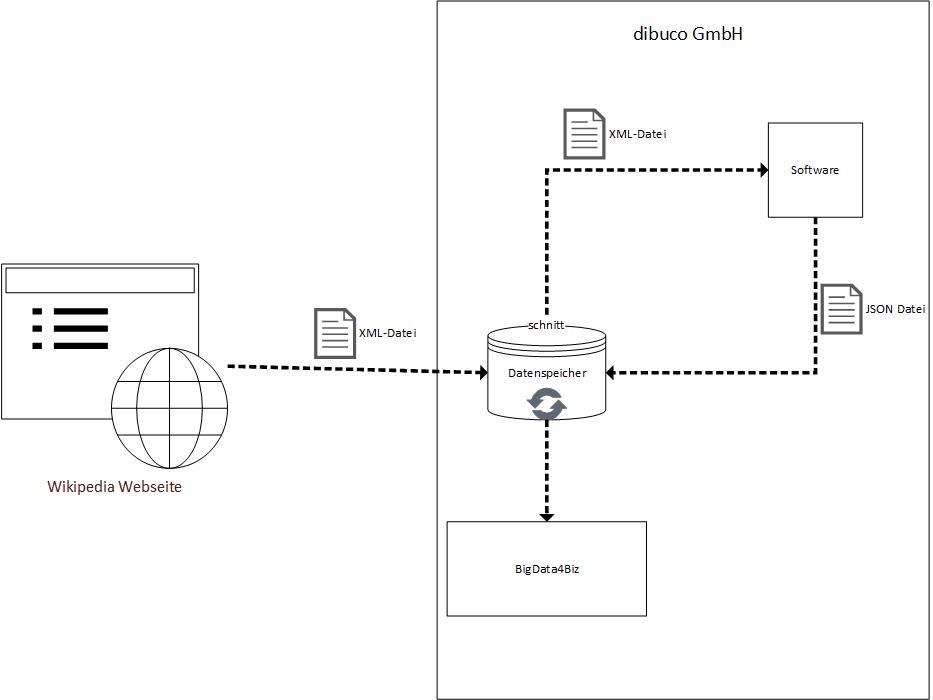


Abbildung 6: Übersicht Mechanismus zur Dokumentähnlichkeitsbestimmung für diese Arbeit.

Die Abbildung 5 zeigt eine Übersicht des Mechanismus zur Dokumentähnlichkeitsbestimmung für diese Arbeit. Daten werden als XML Dokumente aus der Webseite Wikipedia gesammelt, danach erfolgt die Umwandlung von diese in JSON-Dateien. Diese JSON-Dateien werden dann dienen als Basis für den Dokumentenvergleich.

## Versuchsaufbau Für der BayesLSH-Lite Algorithmus

Für die Implementierung des Ansatzes BayesLSH-Lite wurden verschiedenen Elemente genutzt. Unter anderem ein Rechner, die Programmiersoftware R, Daten, den Code.

### Der Rechner

Der für diese Implementierung angewendete Rechner ist ein HP Pavilion Notebook, von der Firma HP hergestellt, das das Betriebssystem Windows 10 Home 64-Bit nutzt, mit einem Mikroprozessor Intel (R) Core(TM) i5-6400U CPU @2.30 GHz. Dieses Notebook verfügt auch über einen 8 GB fähigen Systemspeicher (RAM), eine Systemplatine 82OC 82.30 und ein System-BIOS F.25.

Das HP Pavilion Notebook verfügt auch über zwei Festplattenlaufwerke: Das Laufwerk 1 mit einer Speicherkapazität von 917,21 GB (davon 661,2 GB freier Speicherplatz besteht) und das Laufwerk 2 mit einer Speicherkapazität von 117,26 GB (davon 26,18 GB freier Speicherplatz besteht). Das Laufwerk 1 ist exklusiv für die Speicherung von Dateien (Partition 1) und für Recovery (Partition 2) versehen, während das Laufwerk 2 mit nur eine Partition exklusiv für Programme, Anwendungen und User versehen ist.

Nachdem den angewendeten Rechner beschrieben wurde, ist die nächste Aufgabe die verwendeten Daten zu erläutern.

### Die Daten

**Die Dateien:** die in diese Implementierung angewendete Dateien stammen ursprünglich aus Wikipedia, bzw. wurden als XML-Dateien von Wikipedia aufgenommen und dann mithilfe einer Software in JSON-Dateien umgewandelt um entsprechend für das Projekt BigData4Biz genutzt zu werden. Diese Daten entsprechen dann Entitäten in der BigData4Biz Software. Es wurden dann 90 Dateien im JSON Format genommen für diesen Fall.

Es wurden weiterhin drei unterschiedliche Versuche durchgeführt.

**Versuch 1:** Für den ersten Versuch werden bestimmte Werte gegeben im Code; Für die Berechnung des thresholds die Anzahl an Hashes h =240 und die Anzahl an Bänder b =80 unter der Vorgabe, dass die Anzahl an Hashes teilbar sein mit der Anzahl an Bänder; Für das Corpus wird die Mindestanzahl an zu berücksichtigende n-gramms variieren zwischen 1 und 3.

**Versuch 2:** Für den zweiten Versuch werden bestimmte Werte gegeben im Code; Für die Berechnung des thresholds die Anzahl an Hashes h =200 und die Anzahl an Bänder b=40; Für das Corpus wird die Mindestanzahl an zu berücksichtigende n-gramms variieren zwischen 1 und 3.

**Versuch 3:** Für den dritten Versuch werden bestimmte Werte gegeben im Code; Für die Berechnung des thresholds die Anzahl an Hashes h =40 und die Anzahl an Bänder b=5; Für das Corpus wird die Mindestanzahl an zu berücksichtigende n-gramms variieren zwischen 1 und 3.

### Die Ergebnisse

Die vorher festgelegten Werte für Variable und die Durchführung des BayesLSH-Lite Codes gibt als Rückgabe eine Liste von Dokumenten Paaren mit entsprechenden Ähnlichkeiten. Ergebnisse des genannten Versuchsaufbaus werden in diesem Teil gegeben.

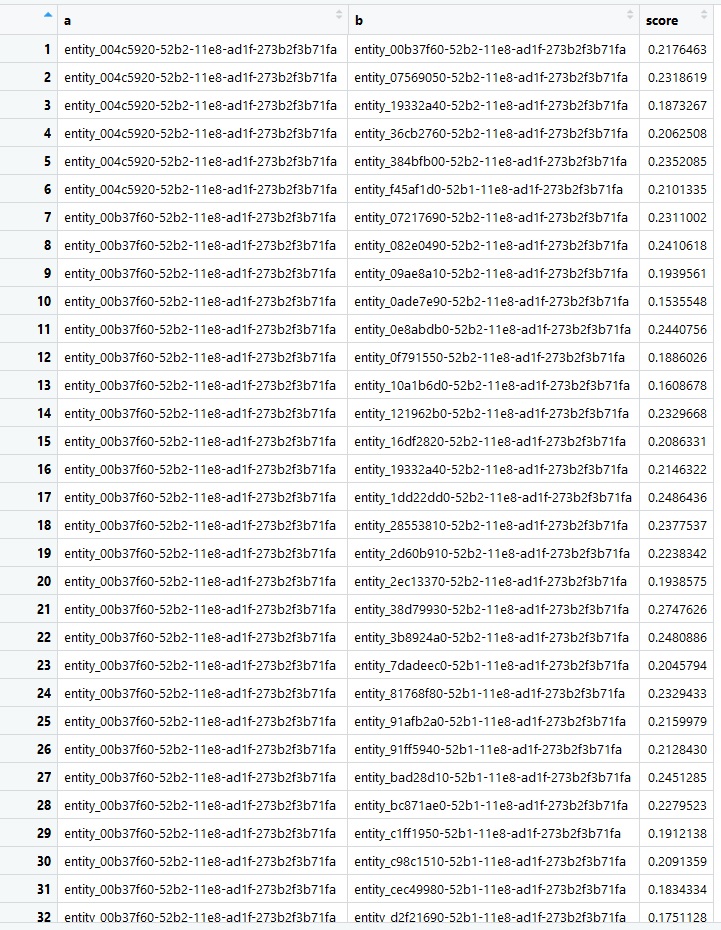


Abbildung 7: Prototyp der zurückgegebene Tabelle des BayesLSH-Lite Algorithmus

Die Abbildung 7 ist ein Prototyp der zurückgegebene Tabelle des BayesLSH-Lite Algorithmus und zeigt eine Liste der Dokumenten Paare mit entsprechenden Ähnlichkeitswerte.

Es werden zunächst Ergebnisse der drei Versuche erläutert.

**Versuch 1:**

* **n = 1, Threshold = 0,232**

Als Ergebnisse wird eine Liste von 820 Dokumenten Paaren mit der entsprechenden Ähnlichkeit zurückgegeben. Auf eine Skala von 0 bis 100, variieren die Ähnlichkeiten zwischen 11 und 33 %. Die Laufzeit beträgt 15 Sekunden.

* **n = 2, Threshold = 0,232**

Als Ergebnisse wird eine Liste von 24 Dokumenten Paaren mit der entsprechenden Ähnlichkeit zurückgegeben. Auf eine Skala von 0 bis 100, variieren die Ähnlichkeiten zwischen 3 und 14 %. Die Laufzeit beträgt 15 Sekunden.

* **n = 3, Threshold = 0,232**

Als Ergebnisse wird eine leere Tabelle ohne Dokumenten Paaren mit der entsprechenden Ähnlichkeit zurückgegeben der threshold ein Wert von 0,232. Die Laufzeit beträgt 15 Sekunden.

**Versuch 2:**

* **n = 1, Threshold = 0,478**

Als Ergebnisse wird eine Liste von 5 Dokumenten Paaren mit der entsprechenden Ähnlichkeit zurückgegeben. Auf eine Skala von 0 bis 100, variieren die Ähnlichkeiten zwischen 18 und 28 %. Die Laufzeit beträgt 10 Sekunden.

* **n = 2,** **Threshold = 0,478**

Als Ergebnisse wird eine leere Tabelle ohne Dokumenten Paaren mit der entsprechenden Ähnlichkeit zurückgegeben. Die Laufzeit beträgt 13 Sekunden.

* **n = 3, Threshold = 0,478**

Es werden die gleichen Ergebnisse wie bei n=2 und Threshold = 0,478 geliefert. Die Laufzeit beträgt 16 Sekunden.

**Versuch 3:**

* **n = 1, Threshold = 0,817**

Als Ergebnisse wird eine Liste von 5 Dokumenten Paaren mit der entsprechenden Ähnlichkeit zurückgegeben. Auf eine Skala von 0 bis 100, variieren die Ähnlichkeiten zwischen 18 und 28 %. Die Laufzeit beträgt 9 Sekunden.

* **n = 2, Threshold = 0,817**

Als Ergebnisse wird eine leere Tabelle ohne Dokumenten Paaren mit der entsprechenden Ähnlichkeit zurückgegeben. Die Laufzeit beträgt 13 Sekunden.

* **n = 3, Threshold = 0,817**

Es werden die gleichen Ergebnisse wie bei n=2 und Threshold = 0,817 geliefert. Die Laufzeit beträgt 14 Sekunden.

## Versuchsaufbau Für der Pairwise Compare Algorithmus

Für den Versuchsaufbau des Pairwise Compare Algorithmus wird der Rechner des Abschnitt 6.1.1 benutzt und es werden dann folgend die Daten, Ergebnisse und Diskussion erläutert.

### Die Daten

**Die Dateien:** die sind die gleichen Dateien, die im Abschnitt 6.1.2 erwähnt wurden. Es wurde nur einen Versuch durchgeführt für den Pairwise Compare Algorithmus.

**Versuch:** Für das Corpus wird die Mindestanzahl an zu berücksichtigende n-gramms variieren zwischen 1 und 3.

### Die Ergebnisse

Der Pairwise Compare Algorithmus liefert Ergebnisse in einer Matrix Form. Die Abbildung 8 ist ein Prototyp der durch den Pairwise Compare Algorithmus gelieferte Matrix und zeigt die Dokumentennamen als Beschriftung der Zeilen und Spalten mit entsprechenden Ähnlichkeiten in Zeilen und den Verweis NA für die Fälle wo keinen Vergleich stattgefunden hat.

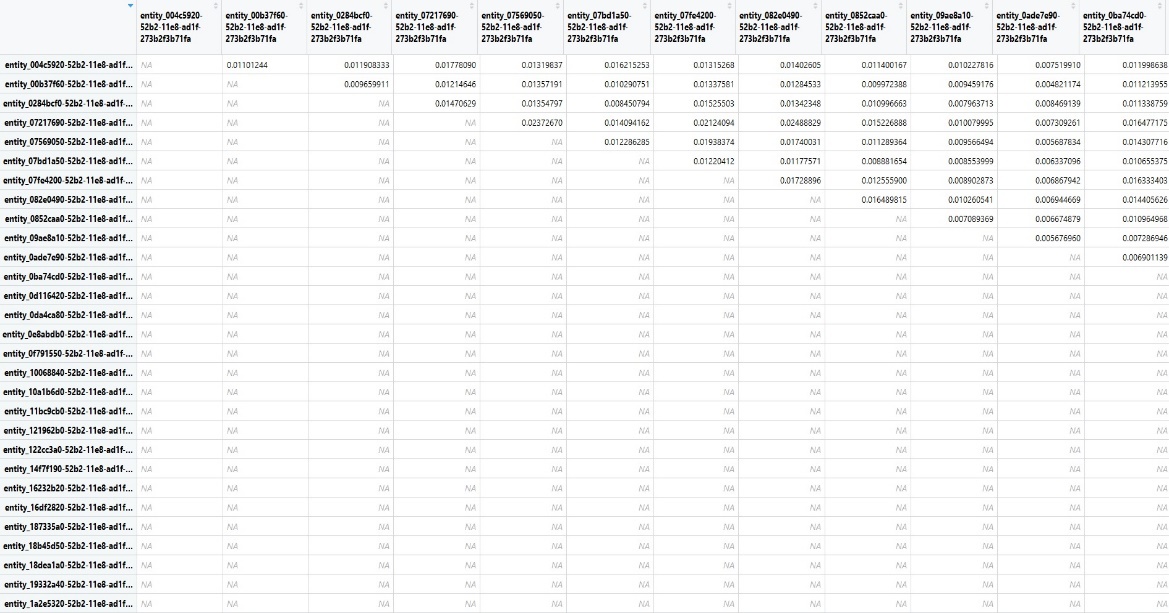


Abbildung 8: Prototyp der durch den Pairwise Compare Algorithmus gelieferte Matrix

Zunächst gilt es die Ergebnisse des Versuches zu erläutern.

**Versuch:**

* **n = 1**

Als Ergebnisse wird eine Matrix mit den Dimensionen 90\*90 von Dokumenten mit der entsprechenden Ähnlichkeit zurückgegeben. Auf eine Skala von 0 bis 100, variieren die Ähnlichkeiten zwischen circa 13 und 28 %. Die Laufzeit beträgt 26 Sekunden.

* **n = 2**

Als Ergebnisse wird eine Matrix mit den Dimensionen 90\*90 von Dokumenten mit der entsprechenden Ähnlichkeit zurückgegeben. Auf eine Skala von 0 bis 100, variieren die Ähnlichkeiten zwischen circa 2 und 9 %. Die Laufzeit beträgt 28 Sekunden.

* **n = 3**

Als Ergebnisse wird eine Matrix mit den Dimensionen 90\*90 von Dokumenten mit der entsprechenden Ähnlichkeit zurückgegeben. Auf eine Skala von 0 bis 100, variieren die Ähnlichkeiten zwischen circa 0 und 1 %. Die Laufzeit beträgt 27 Sekunden.

Ein Vergleich der Ergebnisse der Versuchsaufbauen von BayesLSH und Pairwise Compare Algorithmen wird zunächst durchgeführt.

## Vergleich von beiden Algorithmen

Der Vergleich der beiden Algorithmen wird als Basis die Leistung, die Laufzeit, Effizienz, Skalierbarkeit haben.

### Die Leistung

Bei der BayesLSH-Lite Algorithmus wurden drei Versuche durchgeführt, während bei dem Pairwise Compare Algorithmus nur ein einziger Versuch durchgeführt wurde mit Variation der Werte von n zwischen 1 und 3. Für den Vergleich wurde die CPU-Leistung beobachtet für die jeweiligen Algorithmen, bzw. es wurde die CPU-Leistung für Den BayesLSH-Lite Algorithmus während des Versuchs eins aufgenommen und die CPU-Leistung des Pairwise Compare Algorithmus Versuchs mit Variierung des Werte von n-gramm zwischen 1 und 3. Die Abbildung 9 schildert das Verhältnis zwischen beiden Algorithmen.

Bemerkt wird es für n gleich eins, dass der BayesLSH-Lite Algorithmus eine CPU-Leistung zwischen 18 und 40 % liegt, während bei dem Pairwise Compare Algorithmus CPU-Leistung zwischen 22 und 44% variiert. Weiterhin sind die beide CPU-Leistungen ungleich, haben aber trotzdem die gleiche Prozentspanne. Die CPU-Leistung des Pairwise Compare Algorithmus ist höher als die von dem BayesLSH-Lite Algorithmus.

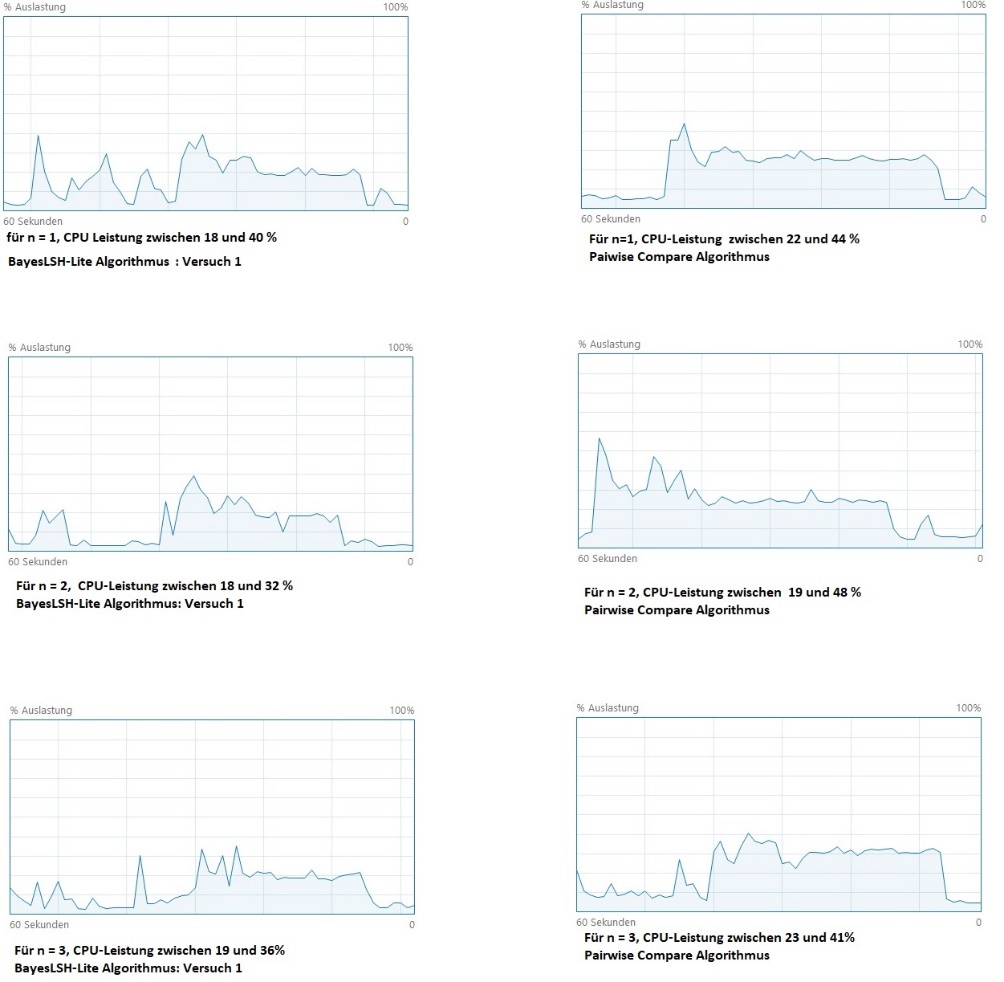


Abbildung 9: Vergleich der CPU-Leistung vom BayesLSH-Lite und Pairwise Compare Algorithmen

Für n = 2, die CPU-Leistung des BayesLSH-Lite Algorithmus liegt zwischen 18 und 32%, während die vom Pairwise Compare Algorithmus zwischen 19 und 48 % liegt. Der Pairwise Compare Algorithmus fordert mehr CPU-Leistung als BayesLSH-Lite auf.

Für n = 3 die CPU-Leistung des BayesLSH-Lite Algorithmus liegt zwischen 19 und 36% und die vom Pairwise Compare Algorithmus zwischen 23 und 41 % liegt. Der Pairwise Compare Algorithmus fordert auch in diesem Fall mehr CPU-Leistung als BayesLSH-Lite auf.

### Die Laufzeit

Beim BayesLSH-Lite Algorithmus für den Versuch 1 beträgt die Laufzeit bei allen Werte von n immer 15 Sekunden; für den versuch 2, bei n= 1, n =2 und n = 3 beträgt die Laufzeit jeweils 10 Sekunden, 13 Sekunden und 16 Sekunden; Beim Versuch 3 beträgt die Laufzeit 9 Sekunden für n =1, 13 Sekunden für n=2 und 14 Sekunden für n= 3. Beim Pairwise Compare Algorithmus beträgt die Laufzeit 26 Sekunden für n = 1, 28 Sekunden für n = 2 und 27 Sekunden für n=3. Es wird bemerkt, die Laufzeit beim Pairwise Compare Algorithmus deutlich höher ist als die vom BayesLSH-Lite Algorithmus.

### Die Effizienz

Bei der Effizienz geht es grob um die gefundene Ähnlichkeit zwischen den Dokumenten. Für n=3 gibt es trotzdem bei dem Pairwise Compare Algorithmus die Rückgabe einer Matrix mit Ähnlichkeiten zwischen 0 und 1 %, während in den drei Versuchen beim BayesLSH-Lite nur eine leere Tabelle zurückgegeben wurde. In Versuch 2 und 3 des BayesLSH-Lite Algorithmus bei n = 1 sind die Ergebnisse gleich, nämlich Ähnlichkeiten liegen zwischen 18 und 28 % und bei Versuch 1 mit n=1 liegen die Ähnlichkeiten zwischen 11 und 30% gegen Ähnlichkeiten zwischen 13 und 28 % bei n=1 Für das Pairwise Compare Algorithmus. Der prozentuale Anteil ist 2% höher beim BayesLSH-Lite und fängt 2% weniger beim Pairwise Compare Algorithmus an. Für n=2 werden Ergebnisse bei BayesLSH-Lite nur für den Versuch 1 zurück geliefert. Die Ähnlichkeit liegt nämlich zwischen 3 und 14% während bei dem Pairwise Compare Algorithmus für n=2 die Ähnlichkeit zwischen 2 und 9% liegt.

### Die Skalierbarkeit

Es bestehen unterschiedliche Skalierbarkeitstypen. Es können unter anderem die Lastskalierbarkeit, Raumskalierbarkeit und die Raum-Zeit-Skalierbarkeit [BON00].

1. Lastskalierbarkeit

Die Raumskalierbarkeit ist für einen Algorithmus bestehend, wenn dieser eine reizvolle Funktionsfähigkeit aufzeigt. Beim BayesLSH-Lite Algorithmus wurden Ergebnisse geliefert abgesehen von den Fällen bei Versuchen wo, bei einer bestimmten Einstellung von n-gramm keine Ergebnisse zurückgeliefert wurden. Der Fall für das Pairwise Compare Algorithmus ist fast gleich, nur ist es so, dass an alle eingestellten Stellen von n Ergebnisse zurückgeliefert wurde. Etwas, das nicht der Fall ist beim BayesLSH-Lite Algorithmus.

1. Raumskalierbarkeit

Diese ist bestätigt, wenn die Speicheranforderungen nicht mehr wachsen als der vom Algorithmus erträglichen Niveau.

Für unsere Algorithmen wurden keine Fehlermeldungen bezüglich des Speichers aufgetaucht.

1. Raum-Zeit-Skalierbarkeit

Die ist gegeben, wenn ein Algorithmus eine ordnungsgemäße Funktionsfähigkeit aufweist trotzt das Wachstum der Anzahl der Objekte.  
Für den BayesLSH-Lite Algorithmus ist die gegeben, da die Anzahl an Objekte auf keine Weise die Funktionsfähigkeit des Algorithmus beeinträchtigt hat.

Für den Pairwise Compare Algorithmus ist die auch gegeben, da dieser problemlos ausgeführt wird.

Nach dem Vergleich der Ergebnisse sowohl von BayesLSH\_Lite Algorithmus als auch von Pairwise Compare wird zunächst eine Diskussion durchgeführt.

## Diskussion der Ergebnisse von beiden Algorithmen

Die im Abschnitte 6.1.3 und 6.2.2 vorgestellten Ergebnisse jeweils für den BayesLSH-Lite Algorithmus und der Pairwise Compare Algorithmus zeigen, dass die Algorithmen unterschiedliche Präsentationsformen zurückgeben, aber den gleichen Zweck haben. Der eine Liefert eine einfache Tabelle und der andere eine Matrix. Aus dem Abschnitt 6.3.2, wird bemerkt, dass derjenige, der eine Liste liefert (BayesLSH-Lite) eine geringe Laufzeit hat als derjenige, der eine Matrix zurückgibt.

Was die Leistung angeht (Abschnitt 6.3.1), der Pairwise Compare Algorithmus fordert mehr CPU-Leistung auf als der BayesLSH-Lite Algorithmus. Es könnte daran liegen, dass der Pairwise Compare Algorithmus mehr Vergleiche für alle Dokumente macht als der BayesLSH-Lite Algorithmus und die Ergebnisse in Matrix Form liefert.

Bezüglich der Effizienz wird gemerkt, dass der Pairwise Compare Algorithmus mindestens ein 0 Prozentualer Ähnlichkeitsgrad mit Ziffer nach dem Komma, während der BayesLSH-Lite Algorithmus mindesten keine Ergebnisse liefern kann. Dieses kann daran liegen, da beim BayesLSH-Lite ein Threshold (Schwellenwert) sowie Bedingungen zum Auswahl von Kandidatenpaare festgelegt wurden. Etwas, das nicht der Fall ist beim Pairwise Compare Algorithmus, wo nur Ähnlichkeit zwischen Dokumenten berechnet wird ohne Angabe eines Grenzwertes oder Grenzbedingung. Die Ergebnisse bei der Matrix sind zwar gut angeordnet aber das lesen der Ähnlichkeitswerte oder die Zuordnung von Dokumentenpaare je nach Ähnlichkeitsgrad ist nicht möglich. Der Benutzer muss darüber hinaus um dieses Problem zu lösen die Matrix in eine Tabelle umwandeln durch Angabe eines Befehls im Code. Im Gegenteil zu den Pairwise Compare Algorithmus der BayesLSH-Lite Algorithmus liefert schon eine überschaubare Tabelle, die dank eines festgelegten Threshold (Schwellenwert) und Bedingungen, relevante Ergebnisse liefert. In diesem Falls Ähnlichkeitswerte, die mindestens bei 3 Prozent liegen werden zurückgegeben.

Was die Skalierbarkeit angeht, können beide Algorithmen als skalierbar angegeben werden, da diese die Bedingungen der Skalierbarkeit entsprechen.

# Zusammenfassung und Ausblick

Die Dokumentähnlichkeitsbestimmung ermöglicht es Mitarbeiter eines Unternehmens schnell Dokumente zu einem bestimmten Thema zu finden und somit den Zeit- und Geldaufwand zu reduzieren durch Maximierung der Speicherbenutzung. Darüber hinaus ist die Erstellung eines Algorithmus zur Dokumentähnlichkeitsbestimmung ist sehr wichtig. Etwas, das das Ziel dieser Arbeit war, nur mit der Spezifizierung eines Algorithmus, der auf Bayesscher Statistik beruht für die Informationsrückgewinnungssoftware BigData4Biz der Firma dibuco GmbH. Voraussetzung dafür war die Auseinandersetzung mit Ansätze zur Dokumentähnlichkeitsbestimmung inklusiv, diejenigen die auf Bayesscher Statistik beruhen. Zwei Ansätze wurden ausgewählt, einen der auf Bayesscher Statistik basiert (BayesLSH-Lite) und anderen, der nicht auf Bayesscher Statistik basiert. Die beiden Algorithmen wurden dann verglichen. Es erfolgte dann die Spezifikation und das Design des BayesLSH-Lite Algorithmus. Im Spezifikationsteil wurden Anforderungen an diesem Algorithmus erläutert und im Designsteil wurde ein Vorlagemuster des BayesLSH-Lite Algorithmus in UML entworfen. Weiterhin wurden der BayesLSH-Lite und der Pairwise Compare Algorithmus implementiert und die entsprechenden Code dazu wurden vorgestellt und erklärt. Es waren für die Implementierung, die in der Spezifikation erläuterten Anforderungen zu berücksichtigen. Danach wurden die beiden Algorithmen evaluiert. Die Evaluation diente dabei die Erfüllung der Anforderungen zu prüfen. Es ging in diesem Fall darum die ausgeführten Experimente und ihren Versuchsaufbauten vorzustellen, die Ergebnisse der Experimente zu präsentieren, ein Vergleich der Ergebnisse jeweiliger Algorithmen durchzuführen und anschließend die Diskussion über die Ergebnisse zu eröffnen und Schlussfolgerungen zu ziehen. Die Anforderungen wurden dann erfüllt und ein wichtiger Evaluationsmerkmal dabei, nämlich die Skalierbarkeit wurde bestätigt für beide Algorithmen. Aus den Ergebnisse kann behauptet werden, dass der BayesLSH-Lite Algorithmus wirksam und effizienter ist als der Pairwise Compare Algorithmus. Es ist somit bestätigt, dass diesen Algorithmus empfehlenswert ist für die Dokumentähnlichkeitsbestimmung.

Durch den praktischen Einsatz dieser Algorithmus in BigData4Biz in der Zukunft kann dieser Algorithmus angepasst, verfeinert und optimiert werden. Es könnten zukünftig noch mehr Algorithmen basierend auf Bayesscher Statistik betrachtet werden und größere Datenbasis für die Experimente genommen werden.

# Abbildungsverzeichnis

[Abbildung 1:High-Level-Architektur von BigData4Biz [DIB18] 13](#_Toc519018560)

[Abbildung 2: Der Pairwise Compare Algorithmus zur Dokumentähnlichkeitsbestimmung [ELO08] 20](#_Toc519018561)

[Abbildung 3: Design BayesLSH-Lite Algorithmus 28](#_Toc519018562)

[Abbildung 4: Der Code zur Implementierung des BayesLSH-Lite mit dem Jaccard Index 32](#_Toc519018563)

[Abbildung 5: Der Code für das Pairwise Comparision Algorithmus 34](#_Toc519018564)

[Abbildung 6: Übersicht Mechanismus zur Dokumentähnlichkeitsbestimmung für diese Arbeit. 35](#_Toc519018565)

[Abbildung 7: Prototyp der zurückgegebene Tabelle des BayesLSH-Lite Algorithmus 37](#_Toc519018566)

[Abbildung 8: Prototyp der durch den Pairwise Compare Algorithmus gelieferte Matrix 39](#_Toc519018567)

[Abbildung 9: Vergleich der CPU-Leistung vom BayesLSH-Lite und Pairwise Compare Algorithmen 41](#_Toc519018568)

# Tabellenverzeichnis

# Quellenverzeichnis

Literaturverzeichnis

|  |  |
| --- | --- |
| [CAR92] | **Manfredo do Carmo**, Riemannian Geometry, Birkhauser Boston (Springer Science+Business Media), 1.Auflage,1992 |
| [HAJ10] | **Christian Haubelt &Jürgen Teich**, Digitale Hardware/Software-Systeme, Springer-Verlag Berlin Heidelberg, 2010 |
| [GHJ96] | **E. Gamma, R. Helm, R. Johnson, J. Vlissides,** Entwurfsmuster, Addison-Wesley (Deutschland) GmbH, 1. Auflage, 1996 |
| [FAM16] | D. Fasel, A. Meier (Hrsg.), Big Data, Edition HMD, Springer Fachmedien, Wiesbaden 2016 |
| [KRU10] | J. Kruschke, Doing Bayesian Data Analysis: A Tutorial with R and BUGS,Academic press,Amsterdam 2011 |
| [BRI99] | R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval, ACM Press, New York 1999. |
| [MRS08] | Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press, New York 2008. |
| [CBB13] | Stefano Ceri, Alessandro Bozzon, Marco Brambilla, Emanuele Della Valle, Piero Fraternali, Silvia Quarteroni, Web Information Retrieval Springer-Verlag Berlin Heidelberg 2013. |
| [MEL15] | Massimo Melucci, Introduction to Information Retrieval and Quantum Mechanics, Springer-Verlag GmbH Berlin Heidelberg 2015. |
| [MET11] | Donald Metzler, A centric view of information retrieval, Springer-Verlag GmbH Berlin Heidelberg 2011 |
|  |  |

Alternatives Literaturverzeichnis

|  |  |
| --- | --- |
| [SAL87] | **Gerard Salton**, Information Retrieval - Grundlegendes für Informationswissenschaftler, McGrawHill, 1987. |
| [SAB88] | **Gerard Salton und Christopher Buckley**, TERM-WEIGHTING APPROACHES in automatic Text retrieval, Pergamon Press plc,1988 |
| [SPA72] | **Sparck Jones**, K. A statistical interpretation of term specificity and its application in retrieval Journal of Documentation, 1972 |
| [ZOU10] | **Zoulfa El Jerroudi**, Eine interaktive Vorgehensweise für den Vergleich und die Integration von Ontologien, Doktorarbeit, EUL Verlag, 2010 |
| [GIN13] | **Mahmoud Gindiyeh**, Anwendung wahrscheinlichkeitstheoretischer Methoden in der linguistischen Informationsverarbeitung, Doktorarbeit, Logos Verlag Berlin GmbH, 2013 (Quelle bei Herangehensweise zur Untersuchung der Dokumentähnlichkeit) |
| [MUL16] | **Lincoln Mullen**, Detect Text Reuse and Document Similarity, Package description,2016 |
| [BON00] | **André B. Bondi**, Characteristics of Scalability and Their Impact on Performance, Paper, 2000 |
| [ELO08] | **Tamer Elsayed, Jimmy Lin, Douglas W. Oard,** Pairwise Document Similarity in Large Collections with MapReduce,Association for computational linguistic, 2008 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Verzeichnis der Internetquellen

|  |  |
| --- | --- |
| [WUA97] | **Alexander Kaiser**, Die Automatische Indexierung, <https://ai.wu.ac.at/~kaiser/diss/node28.html> (Abruf:07.07.2018) |
| [GAB17] | **Oliver Bendel,** Big Data, <http://wirtschaftslexikon.gabler.de/Definition/big-data.html>  (Letzter Abruf:10.07.2018) |
| [COM18] | Computer Bild, <http://www.computerbild.de/artikel/cb-Ratgeber-Internet-So-schuetzen-Sie-sich-vor-der-E-Mail-Spam-3574047.html> (Letzter Abruf:05.02.2018) |
| [LER17] | Lern Plattform, <https://www.lernhelfer.de/schuelerlexikon/mathematik-abitur/artikel/thomas-bayes> (Letzter Abruf: 30.12.2017) |

# Index

Aliquam 8

Index 8

Lorem ipsum 8

Maecenas 8

Phasellus 8

Sperrvermerk 14

Stichwortverzeichnis 8

vulputate 8

# Sperrvermerk

An dieser Stelle kann ein Sperrvermerk, sofern vorhanden, begründet und zeitlich eingegrenzt werden.

# Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet habe. Die Arbeit wurde noch keiner Kommission zur Prüfung vorgelegt und verletzt in keiner Weise Rechte Dritter.

Reutlingen den, 31.07.2018

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 (Unterschrift)