1. Die Informationsrückgewinnung-Middleware-Lösung: BigData4Biz
   1. Vorstellung

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 Datensee. 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.

* 1. Anforderungen

BigData4Biz muss bestimmte Eigenschaften haben und eine bestimmte Leistung erbringen. Für BigData4Biz es bestehen unterschiedliche Anforderungen, nämlich funktionelle Anforderungen, nicht funktionelle Anforderungen, Plattform Anforderungen, Ähnlichkeitsanforderungen, Operationsanforderungen und Geschäftsanforderungen [DIB18].

* + 1. 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 von den ursprünglichen Wert 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 muss vermieden werden.

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

* + 1. 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.

* + 1. 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 von 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.

* + 1. 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 Entität-Backlink 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. Falls der Ausgleich des Ausfalls von einer der Ausdauer erfolglos ist, musste die Plattform benachrichtigt werden über den inkonsistenten Zustand der Entität.

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. Abschnitte rouges a relire

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.

* + 1. Anforderungen an dem 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.

* 1. 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.

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.

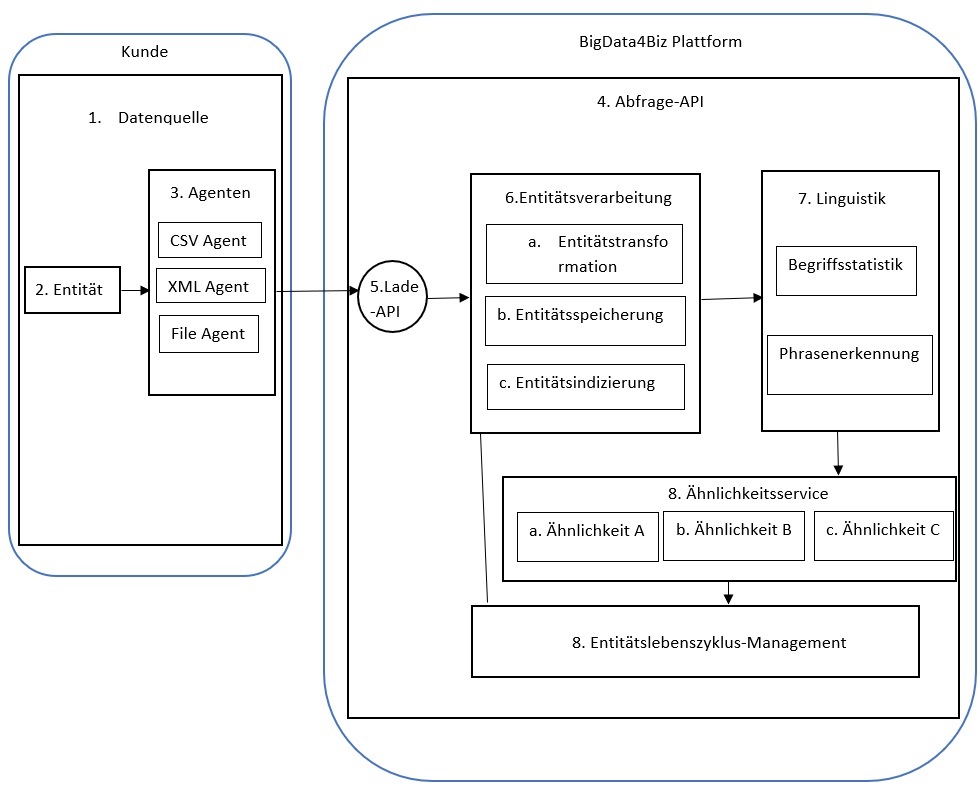


Abbildung 1: High-Level-Architektur von BigData4Biz [DIB18].

1. Die Datenquelle (1) 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 (2) und Agenten (5), die wichtig sind für die spätere Ausnutzung von Daten in BigData4Biz. Für die Datenquelle die Ausgabe von Entitäten verschiedener Klassifizierungen und Strukturen ist möglich.
2. Die Entität (2) 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 (2) durch Agenten (3) umgewandelt um später in BigData4Biz über eine Lade-API (5) 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.
3. Die Übertragung der Entitäten (2) von einer Datenquelle (1) 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 (5) gibt, deren erwarteten Aufgabe den Empfang von Entitäten (2) ist. Die Entitätsagenten (3) 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 (3) führt die Umwandlung der extrahierten Daten in die Entitätsform selbst durch. Der Entitätsagent (3) befindet sich in der Sicherheitszone der Datenquelle (1) zur effektiven Erkennung der Datenänderungen und einfachere Sendung von Daten. Es bestehen schon definierte Standardagenten für allgemeine physische Datenquelle (1), die den vollständigen ETL-Schritt liefern. Die Entitätsagenten (3) sind Buchhalter aller aus ihrer Datenquelle (1) extrahierte Entitäten. Damit können Datenänderungen an Entitäten in BigData4Biz verfolgt werden und die Entität kann zum Zeitpunkt des Ladens aktualisiert werden.
4. Abfrage-API (4) 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.
5. Die Lade-API (5) entspricht ein REST-konformer Dienst von BigData4Biz, die für den Empfang und die Ladung von Entitäten (2) zur weiteren Verarbeitung in BigData4Biz dient.
6. Die Entitätsverarbeitung (6) ist der Teil der Architektur, der sich um die Entitätsversorgung kümmert in BigData4Biz nachdem diese über die Lade-API (5) 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 (5.a), die Entitätsspeicherung (5.b) und die Entitätsindizierung (5.c). Die Entitätsindizierung (5.c) zum Beispiel ist ein wichtiger Prozess, der die Assoziation eines Vokabulars aus Schlüsselwörtern und allen Dokumenten eines Textkorpus durchführt.
7. Die Linguistik (7) 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 (4), 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.
8. Die Ähnlichkeitsdienste (8) 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 (8) verfügen über Ähnlichkeitsalgorithmen, die die Berechnung von durch Prädikaten ausgedruckte Ähnlichkeiten durchführen. Die Kernähnlichkeitsalgorithmen von BigData4Biz haben als Basis linguistischen Statistiken (7.a) wie TF-IDF. Die Ähnlichkeitsdienste (8) sind unabhängig von den anderen Diensten und benutzen ihre eigene Persistenz zur effektiven Berechnung von SPO-Beziehungen[DIB18].
9. Das Entitätslebenszyklusdienst (9) ist für die Überwachung von jeden schritten der Entitätsverarbeitung (6) zuständig. (Kompletter Teil noch zu ändert: die Nummerzuweisungen wurden geändert [Siehe Abbildung Word Datei])
   1. he

# Abkürzungsverzeichnis

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

[DIB18] <https://dibuco.atlassian.net/wiki/spaces/CI/pages/16291188/Product+-+Specification> (Letzter Zugriff:08.05.2018)

[SHR11] G. Starke, P. Hruschka, Software-Architektur kompakt, Spektrum Akademischer Verlag, Heidelberg 2011.

[VAC09] O. Vogel, I. Arnold, A. Chughtai, E. Ihler, T. Kehrer, U. Mehlig, U. Zdun, Software-Architektur, Spektrum Akademischer Verlag, Heidelberg 2009.