Michael Oettinger

Data Science und AI

Eine praxisorientierte Einführung im Umfeld von Machine Learning, künstlicher Intelligenz und Big Data

3. erweiterte Auflage

Druck und Distribution im Auftrag des Autors:
tredition GmbH, Heinz-Beusen-Stieg 5, 22926 Ahrensburg, Germany

Inhalt

1 Einleitung

Ich habe im Jahr 2017 die erste Auflage dieses Buches geschrieben, weil ich meinen Job im Softwarevertrieb für ein kleineres US-Softwareunternehmen gekündigt hatte und mich als Data-Scientist selbstständig machen wollte. Ich hatte keine Referenzen und Kunden und dachte mir, dass man einem Autor eines Fachbuches die Kompetenz für das Fachgebiet unterstellen kann und ich so den Eintritt in die Welt der Freelancer und vor allem in meine ersten Projekte finden kann.

Spoiler: Es hat funktioniert und ich arbeite seit nunmehr sieben Jahren in unterschiedlichen Projekten für unterschiedliche Kunden (übrigens bei Vollauslastung und vor allem bei 'Vollzufriedenheit').

Nach drei Jahren war es Zeit für ein Update, sodass im Jahr 2020 eine zweite aktualisierte Auflage des Buches veröffentlicht wurde.

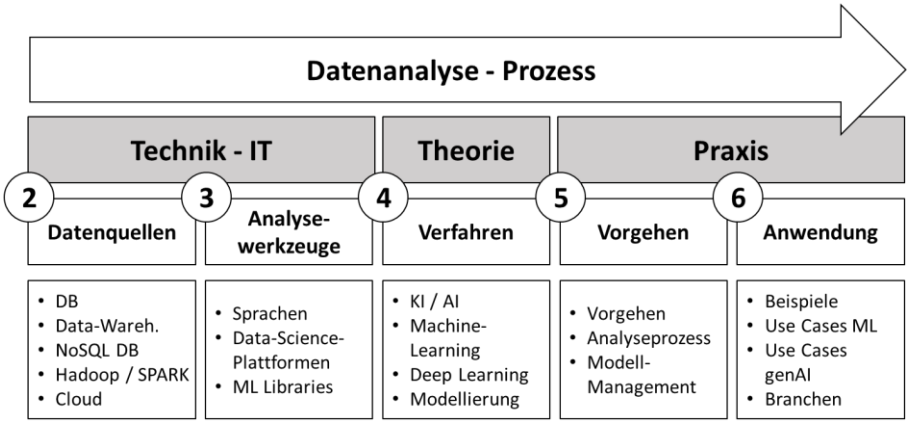
Die Gründe für die Erstellung der hier vorliegenden dritten Auflage sind:

- Data-Science und vor allem die dazugehörende Softwaretechnologie haben sich weiterentwickelt.
- Spätestens mit der Veröffentlichung von ChatGPT ist das Thema künstliche Intelligenz in aller Munde und eine Einordnung von Data-Science, Machine Learning und Artificial Intelligence scheint dringend notwendig.
- Mit der Projekterfahrung der letzten sechs Jahre würde ich das Buch heute anders gestalten. Die Grundstruktur und der Aufbau passen, aber an vielen Stellen würde ich heute deutlichere Aussagen machen. Weniger Abwägen und wissenschaftliches Erarbeiten und mehr deutliche, pragmatische Empfehlungen geben.
  Also habe ich mich hingesetzt und das Buch aufgefrischt. Wenn man so will, ist es nun polemischer geworden, da ich in allen Kapiteln 'meinungsstarke' und deutliche Kommentare eingefügt habe. Aber es ist auch pragmatischer, da es nun Code-Beispiele in Python bzw. SQL enthält und einige Cheat-Sheets angefügt wurden.

Bedanken möchte ich mich an dieser Stelle bei allen Kunden und Kollegen, die mir das Vertrauen geschenkt und mir so ermöglicht haben, als Data Scientist zu arbeiten. Außerdem möchte ich mich bei meiner Frau und meiner Tochter bedanken. Ohne meine Familie wäre zwar dieses Buch wahrscheinlich einen oder zwei Monate früher fertig geworden, aber mein Leben wäre sinnloser gewesen.

Gliederung des Buches

Das Buch ist folgendermaßen gegliedert:



Nach der Einführung in **Kapitel 1** orientiert sich die Gliederung des Buches am Prozess der Datenanalyse. Von der Datenquelle geht es über die verwendeten Werkzeuge und die eingesetzten Verfahren bis hin zum konkreten Vorgehen und Beispielen in der Praxis.

Kapitel 2 enthält Erläuterungen zu den ‘Datentöpfen‘ aus einer technischen Perspektive. Wo und wie werden die Daten bereitgestellt, die als Quelle für die Datenanalyse herangezogen werden? Konkret werden die am weitesten verbreiteten Arten von Datenbanken vorgestellt:

- Flatfiles,
- ODBC-Datenbanken,
- Data-Warehouse,
- NoSQL-Datenbanken,
- Hadoop- und Spark-Plattformen sowie
- Cloud-Speicher.

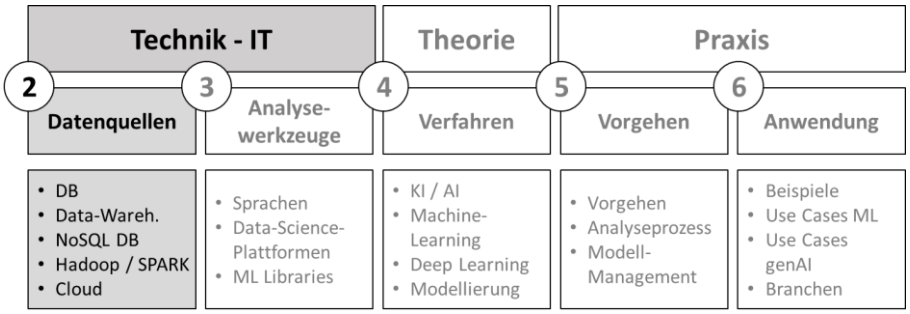
In **Kapitel 3** wird auf die Werkzeuge – d. h. die Softwarelösungen – eingegangen, mit denen die Daten analysiert werden. Dabei wird zwischen den wichtigsten Sprachen (SQL, Python, R), den Data-Science-Plattformen und den Machine Learning-Librarys unterschieden. Zu diesen Softwareanwendungen gibt es sowohl Open-Source- als auch kommerzielle Angebote.

In **Kapitel 4** wird auf die gebräuchlichsten Analyseverfahren eingegangen. Dabei handelt es sich um Verfahren aus den Bereichen Statistik, Mathematik, Machine Learning, künstliche Intelligenz und Computer-Science. Es wird versucht, die Verfahren zu strukturieren und im Einzelnen so darzustellen, dass ein Grundverständnis für ihre Möglichkeiten und Grenzen aufgebaut werden kann.

Kapitel 5 ist der Praxis gewidmet, indem erläutert wird, wie Analytics-Projekte in Unternehmen oder Forschungseinrichtungen durchgeführt werden. Die bewährten Vorgehensmodelle werden vorgestellt. Außerdem wird auf das Thema Modellmanagement eingegangen. Dies ist vor allem dann wichtig, wenn in größeren Teams zusammengearbeitet wird und über die Zeit eine Vielzahl von Analysemodellen erstellt, getestet, angepasst und wieder verworfen wird.

In **Kapitel 6** werden Use-Cases – d. h. Anwendungsfälle – für die besprochenen Verfahren und Techniken vorgestellt. Dabei geht es nicht nur um konkrete Einzelfälle, sondern auch um den Versuch, ein Bild über mögliche Einsatzszenarien zu geben. Die Use-Cases werden vorgestellt und die Besonderheiten ausgewählter Branchen werden diskutiert.

2 Daten bereitstellen



Data-Science bezeichnet den Prozess, durch die Analyse von Daten mit geeigneten Verfahren Erkenntnisse zu gewinnen. Die erste Frage, die sich stellt, ist diejenige nach der Quelle der Daten. Woher kommen die zu analysierenden Daten und wo und wie werden sie bereitgestellt? Im Folgenden wird auf diese *Datenquellen* näher eingegangen. Konkret handelt es sich dabei um:

- Flatfiles
- Relationale Datenbanken
- Data-Warehouses
- NoSQL-Datenbanken
- Hadoop
- Cloud-Datenbanken

2.1 Flatfiles

Die einfachste Form der Datenbereitstellung sind Flatfiles, also Tabellen und strukturierte Textdateien, die man aus operativen Systemen wie z. B. ERP-Systemen exportiert oder über Befragungen gewonnen hat. Die Dateien werden in unterschiedlichen Formaten zur Verfügung gestellt. Die gebräuchlichsten sind:

- csv
- xls
- xml
- produktspezifische Formate (SPSS, SAS, Stata, ARFF, DBase …)

Bei dieser Form der Datenanalyse handelt es sich meist nicht um ‘Big Data’ (auch wenn die Größe der Files grundsätzlich nahezu unbegrenzt sein kann), aber dennoch spielen Flatfiles nach wie vor eine wichtige Rolle in Data-Science-Projekten. Es muss z.B. kein Zugang zur Datenbank eines Produktivsystems eingerichtet werden, was meist einen höheren Aufwand im Bereich Berechtigungen und Netzwerkzugang bedeutet. Stattdessen werden die Daten aus dem Quellsystem exportiert und dann in das Analysesystem eingelesen, wo die eigentliche Analyse bzw. Modellierung stattfindet. Liegt eine sehr hohe Anzahl an Flatfiles vor, bietet es sich an, den Prozess des Einlesens und Zusammenfassen der Daten z. B. durch ein Programm in Python zu automatisieren.

2.2 Relationale Datenbanksysteme

Relationale Datenbanksysteme dienen der Datenverwaltung und beruhen auf einem tabellenbasierten, relationalen Datenbankmodell. Sie werden auch als RDBMS (Relational Database Management System) bezeichnet. Zum Abfragen und Manipulieren der Daten wird überwiegend die Datenbanksprache SQL (Structured Query Language) eingesetzt.

Relationale Datenbanken folgen einem grundsätzlichen Schema. Daten werden in Tabellen gespeichert, wobei die Spalten die Variablen darstellen und die Zeilen die einzelnen Datensätze. Datenbanken werden dadurch ‘relational’, dass es Relationen – also Verbindungen – zwischen den Tabellen gibt. Diese werden eingeführt, um eine redundante Speicherung der gleichen Daten

zu vermeiden. Damit wird Speicherplatz gespart und inkonsistente Datenhaltung vermieden. Beispielsweise werden bei einer Datenbank für *Kunden* nicht für jeden einzelnen Kunden die Unternehmensdaten angegeben, sondern die Kategorie *Unternehmen* wird als eigenständige Tabelle ausgelagert und über eine Relation den einzelnen Kunden zugeordnet. Ändert sich etwas an der Adresse des Unternehmens, muss dies nur an einer Stelle geändert werden – durch die Relation wird den einzelnen Kunden automatisch das entsprechende Unternehmen zugeordnet.

Tabelle	Kunde	
Name	**Vorname**	**Unternehmen**
Karl	Mustermann	U1
Peter	Müller	U2
Claudia	Maier	U1
…		

Tabelle	Unternehmen		
UntNr	**Unternehmen**	**Strasse**	**Ort**
U1	ACME	Goethestr. 1	Berlin
U2	Müller GmbH	Haupttsr. 2	Hamburg
U3	ABC AG	Schillerstr. 1	Essen
…			

Trotz neuerer Entwicklung (siehe den folgenden Abschnitt) stellen relationale Datenbanken nach wie vor die große Mehrzahl der Datenspeicher in Unternehmen dar und sind zentraler Bestandteil der meisten operativen Anwendungen (ERP, CRM, HCM, SCM, Fachsysteme …).

Die wichtigsten Anbieter sind:

- Oracle (Marktführer nach Umsatz)
- Microsoft SQL Server (Marktführer in bestimmten Märkten und auf bestimmten Plattformen)
- MySQL (Open Source, von Oracle erworben, höchste Anzahl an Implementierungen)
- PostgreSQL (Open Source)
- IBM DB2
- SAP Adaptive Server / SQL Anywhere / SAP MaxDB
- Amazon RDS (Cloud-Angebot für RDBS)

2.3 Data-Warehouse

Ein Data-Warehouse (DW oder DWH) ist eine zentrale Sammlung von Daten, die sich aus verschiedenen Quellen speist und vor allem für den Zweck der Analyse und der betriebswirtschaftlichen Entscheidungshilfe dauerhaft gespeichert wird.

Meistens wird ein Data-Warehouse aus zwei Gründen aufgebaut:

- Es soll eine **Integration** von Daten aus verteilten und unterschiedlich strukturierten Datenbeständen erfolgen. Im Data-Warehouse können die Daten konsistent gesichtet und datenquellenübergreifend ausgewertet werden. Die zeitaufwendigen und technisch anspruchsvollen Aufgaben der Datenextraktion und -integration aus verschiedenen Systemen erfolgt damit (im Ideal) einmalig und an zentraler Stelle. Die Daten stehen dann für Analysen und Reporting für die Fachabteilungen ‘konsumbereit’ zur Verfügung.
- Durch eine **Trennung** der (oft ‘sensiblen’) Daten in den operativen Systemen von den für das Reporting genutzten Daten im Data-Warehouse soll sichergestellt werden, dass durch die Datenabfragen für Analysen und Reporting keine operativen Systeme ‘gestört’ werden. Niemand möchte, dass der Azubi in der Vertriebsabteilung durch eine Abfrage der kompletten, weltweiten Produktverkäufe, nach Wochen und Postleitzahl gegliedert, das Buchhaltungssystem für eine halbe Stunde lahmlegt.

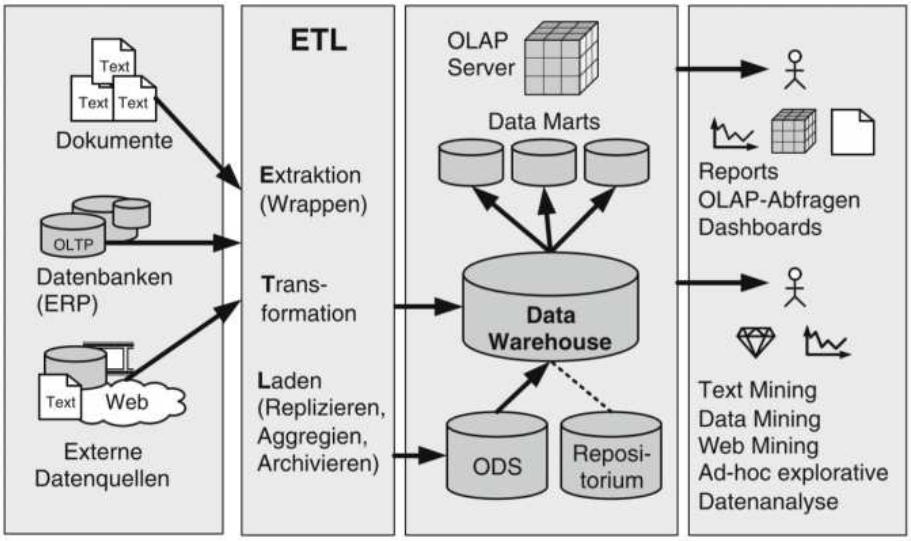


Abbildung 1: Architektur eines Data Warehouses (Müller & Lenz, 2013, S. 19)

Technisch gesehen sind Data-Warehouse-Systeme eine Sammlung von Softwarekomponenten, die die Umsetzung des Data-Warehouse-Konzeptes ermöglichen. Sie bestehen aus:

- **ETL-Komponenten**, die den ETL-Prozess (also die Extraktion, Transformation und das Loading der Daten) unterstützen,
- dem **Core-Data-Warehouse**, also einer Sammlung von gemanagten Datenbanksystemen, die auf Parallelisierung und Performance für das Handling riesiger Datenmengen optimiert sind,
- den 'vorbereiteten' **Aggregationen** (Star-Schemas), die Auswertungen beschleunigen.
- einem **User Interface**, das die Verwaltung und die Auswertung der Datenbestände ermöglicht.

Die wichtigsten Anbieter von Data-Warehouse-Systemen sind:

- Oracle
- Teradata

- Microsoft
- IBM
- SAP

Data Lake

In letzter Zeit wird immer häufiger der Begriff ‘Data Lake’ verwendet. Es handelt sich dabei um ein Konzept, das als eine Erweiterung des Data-Warehouse-Gedankens gesehen werden kann, der dann aber technisch mit Hadoop- oder NoSQL-Mitteln umgesetzt wird (siehe die folgenden zwei Abschnitte).

Im Unterschied zum Data-Warehouse, wo die Daten aus verschiedenen Quellen bezogen und dann so aufbereitet werden, dass sie vergleichbar sind und damit aggregiert werden können (ETL-Prozess), werden beim Data Lake die Daten erst einmal im ursprünglichen Format und unbearbeitet gesammelt. Eine Bearbeitung bzw. Transformation der Daten erfolgt dann erst bei Bedarf vor der eigentlichen Analyse (ELT-Prozess). Diese Vorgehensweise eignet sich also vor allem für

- eher unstrukturierte Daten, z. B. aus sozialen Medien, Blogbeiträgen, Bild- und Videodateien,
- strukturiertere XML- bzw. HTML-Daten,
- oder für Sensor-Daten.

Damit sind wir nun wirklich im Bereich Big Data angekommen. Die große Herausforderung ist es an dieser Stelle, diesen erstmal unbearbeiteten ‘Datensee’ tatsächlich für Analysen und damit einhergehend für den Erkenntnisgewinn zu nutzen. Ein Datentümpel, der ständig mit unnützen Datenmengen ergänzt wird und wächst und wächst, ist wertlos.

Die klassischen Analyseverfahren (siehe Abschnitt 4.5) sind für strukturierte Daten konzipiert. Eine Analyse der unstrukturierten Daten setzt also voraus, dass diese in irgendeiner Form strukturiert werden, um sie im Anschluss mit

den vorhandenen Verfahren analysieren zu können. Nur durch eine integrierte Datenstrategie, die die strukturierten und unstrukturierten Daten miteinbezieht, können die Schätze des Big Data tatsächlich gehoben werden.

2.4 NoSQL

Unter dem Begriff NoSQL werden unterschiedliche Arten von Datenverwaltungssystemen zusammengefasst. Ganz wichtig vorneweg: NoSQL steht **nicht** für ‘no SQL’, also ‘kein SQL’! Das ‘No’ bedeutet vielmehr ‘not only’. NoSQL ist also keine Anti-SQL-Bewegung, sondern soll eine Alternative bzw. Bereicherung zur SQL-Welt darstellen.

Den unterschiedlichen Ausprägungen von NoSQL-Datenbanken ist gemeinsam, dass sie für Anwendungsfälle geschaffen wurden, in denen die verfügbaren SQL-basierten Datenbanken an ihre Grenzen stießen und daher nicht oder nur mit sehr großem Aufwand einsetzbar waren.

Die Architektur vieler NoSQL-Datenbanken setzt auf den Einsatz einer großen Anzahl kostengünstiger Rechnersysteme zur Datenspeicherung, wobei die meisten Knoten gleichrangig sind. Eine Skalierung erfolgt dann einfach durch Hinzufügen weiterer Knoten.

NoSQL-Datenbanken unterscheiden sich hinsichtlich der Art der ‘Verschlüsselung’. Es gibt ‘Key-Value-Stores’ oder komplexere, dokumentenorientierte Ansätze, die zusätzlich zu Dokumenten noch Verknüpfungen zwischen Dokumenten bieten.

NoSQL-Datenbanken werden vor allem dann eingesetzt, wenn SQL-Datenbanken an ihre Grenzen stoßen. In NoSQL-Systemen lassen sich z. B. wesentlich größere Mengen an Daten performant ablegen und aufrufen. Bei komplexen Abfrageanforderungen, etwa im Bereich unstrukturierter Daten wie Video-, Audio- oder Bilddateien, erlauben einige NoSQL-Datenbanken baumförmige Strukturen der Metadaten ohne ein fest definiertes Datenschema und deren flexible Abfrage. Bei Daten mit schwankendem Typ und Inhalt eignen

sich NoSQL-Datenbanken besser, weil sich die Daten nicht länger in das ‘SQL-Korsett’ von Tabellen und Relationen pressen lassen müssen.

Man muss sich aber bewusst darüber sein, dass die Verfahren, mit denen aus Daten Erkenntnisse für eine Prognose gewonnen werden, auf strukturierte Daten angewiesen sind. Das bedeutet nicht, dass das ‘SQL-Korsett’ für die Rohdaten eingehalten werden muss, aber die Aufbereitung vor der Analyse erfordert eine Strukturierung. Bei der Verwendung von NoSQL-Datenbanken müssen daher die ja immer vorhandenen Strukturen der Datenhaltung beachtet und die entsprechende Aufbereitungsschritte angewendet werden.

Wichtige Anbieter von NoSQL-Datenbanken sind:[1]

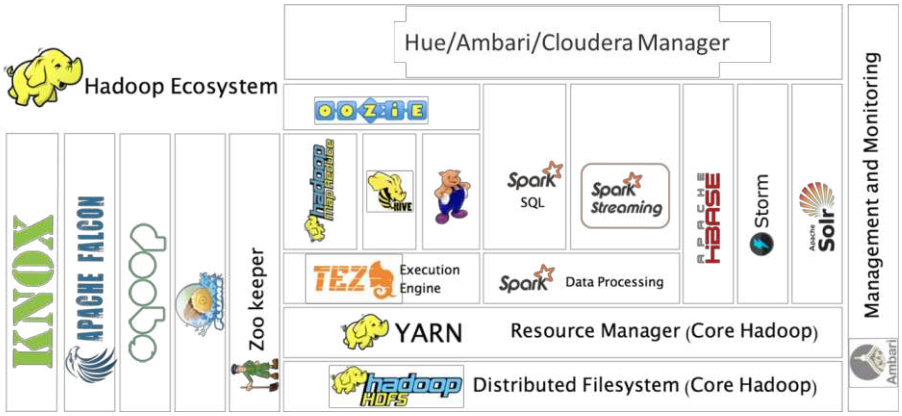
- MongoDB
- Cassandra
- Redis
- HBase
- Couchbase
- NoSQL-Angebote der Cloudanbieter wie AWS und MS Azure

2.5 Hadoop/Spark

Apache Hadoop ist ein Software-Framework, mit dessen Hilfe rechenintensive Prozesse mit großen Datenmengen auf Server-Clustern bearbeitet werden können. Anwendungen können mit der Unterstützung Hadoops komplexe Aufgaben auf Tausende von Rechnerknoten verteilen und Datenvolumina im Petabyte-Bereich verarbeiten. Es basiert ursprünglich auf dem MapReduce-Algorithmus und Grundideen des Google-Dateisystems. Hadoop wird von der Apache Software Foundation – einer Gemeinschaft von Entwicklern, die Open-Source-Softwareprodukte entwickeln – als Top-Level-Projekt vorangetrieben.

[1] Vgl.: http://nosql-database.org/

Hadoop besteht aus vier Kernmodulen und weiteren Komponenten, die zum Hadoop Ecosystem gerechnet werden.



Die vier Kernmodule sind:

- **Hadoop Common**: Hilfswerkzeug, das die Hadoop-Komponenten verwaltet bzw. unterstützt.
- **Hadoop Distributed File System** (HDFS): HDFS ist ein hochverfügbares Dateisystem zur Speicherung sehr großer Datenmengen auf den Dateisystemen mehrerer Rechner (Knoten). Dateien werden in Datenblöcke mit fester Länge zerlegt und redundant auf die teilnehmenden Knoten verteilt. Dabei gibt es Master- und Slave-Knoten. Ein Master-Knoten, der sogenannte NameNode, bearbeitet eingehende Datenanfragen, organisiert die Ablage von Dateien in den Slave-Knoten und speichert anfallende Metadaten. HDFS unterstützt dabei Dateisysteme mit mehreren 100 Millionen Dateien.
- **Hadoop YARN**: Eine Softwarelösung, die die Verwaltung der Ressourcen (also das Job-Scheduling) eines Clusters übernimmt.

- **Hadoop MapReduce**: Ein auf YARN basierendes System, das paralleles Prozessieren großer Datenmengen realisiert. Hadoop beinhaltet den MapReduce-Algorithmus, dieser gilt aber zunehmend als veraltet und wird durch graphenbasierte Verfahren (Spark, Tez) ersetzt.

Insbesondere **Spark** hat mittlerweile die größere Verbreitung im Hadoop-Umfeld und hat MapReduce als Prozess-Engine abgelöst.

Im Rahmen von Apache werden weitere Projekte als zum **Hadoop Ecosystem** zugehörig gezählt:

- **Ambari**: Ambari ist eine Managementplattform, die die Verwaltung (Provisionierung, Management, Monitoring) der Hadoop-Cluster vereinfachen soll. Unterstützt werden: HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop.
- **Avro**: Avro ist ein System zur Serialisierung von Daten.
- **Cassandra**: Cassandra ist ein skalierbares NoSQL-Datenbanksystem für Hadoop-Cluster.
- **Chukwa**: Chukwa ermöglicht die Datensammlung und Echtzeitüberwachung sehr großer verteilter Systeme.
- **HBase**: HBase ist eine skalierbare Datenbank zur Verwaltung großer Datenmengen innerhalb eines Hadoop-Clusters. Die HBase-Datenbank basiert auf Googles BigTable. Diese Datenstruktur ist für Daten geeignet, die selten verändert, dafür aber häufig ergänzt werden. Mit HBase lassen sich Milliarden von Zeilen verteilt und effizient verwalten.
- **Hive**: Hive ist eine Data-Warehouse-Infrastrukturkomponente, die Hadoop-Cluster um Data-Warehouse-Funktionalitäten erweitert. Mit HiveQL wird eine SQL-Sprache zur Abfrage und Verwaltung der Datenbanken bereitgestellt.
- **Mahout**: Mahout ist eine skalierbare Machine Learning- und Data-Mining-Library, die aber nicht mehr weiterentwickelt wird.

- **Pig**: Pig ist einerseits eine Hochsprache für Datenfluss-Programmierung, andererseits ein Framework, das die Parallelisierung der Rechenvorgänge unterstützt.
- **Spark**: Spark ist eine performante In-Memory-Batch-Prozess-Engine für Hadoop-Daten. Spark unterstützt ETL-, Machine Learning-, Streaming- und Graphenprozesse.
- **Tez**: Apache Tez ist ein allgemeines Datenfluss-Programmier-Framework. Die ursprünglich von Hortonworks entwickelte Anwendung unterstützt Directed Acyclic Graph (DAG). Tez baut auf YARN auf und wird auch durch YARN gesteuert. Tez kann jeden MapReduce-Job ohne Modifikationen ausführen. MapReduce-Jobs können in einen Tez-Job überführt werden, was die Leistung steigert.
- **ZooKeeper**: ZooKeeper ist ein performantes System zur Koordination und Konfiguration verteilter Systeme.

Aus der Aufzählung und kurzen Beschreibung der Hadoop-Komponenten wird deutlich, dass es sich bei Hadoop nicht um ein einfaches Datenmanagement-Tool handelt. Es ist vielmehr ein komplexes und sich dynamisch veränderndes Sammelsurium an Projekten und Softwareprodukten, die der Idee der verteilten Datenhaltung von Big Data folgen.

Die unterschiedlichen Hadoop-Komponenten können von der Homepage der Apache Foundation kostenlos heruntergeladen werden. Unternehmen greifen aber bei Hadoop auf die Dienstleistungen kommerzieller Hadoop-Distributoren zurück. Diese bieten vorgefertigte Pakete mit z. T. zusätzlichen Komponenten an. In der Regel fallen keine Lizenzkosten an, es wird aber eine Subscription-Fee verlangt, also eine Mietgebühr für die Wartung, Pflege und den Support der Software. Wichtige Anbieter sind:

- Cloudera (Fusioniert mit Hortonworks im Q1 2019)
- MapR
- IBM
- Pivotal