Zweiter Frühling für die Analyse unstrukturierter Daten

Christopher Thomsen, OPITZ CONSULTING GmbH

Die meisten weltweit verfügbaren Daten sind unstrukturiert, die Mehrheit dieser unstrukturierten Daten sind Texte. Doch die meisten beschreibenden und vorhersagenden Analyse-Verfahren können den Informationswert dieser Daten nicht nutzen. Dabei können vor allem textuelle Daten aus sozialen Medien, Mails, Geschäftsberichten, Webseiten-Inhalten, Fachartikeln und den vielen weiteren Quellen von textbasierten Informationen dabei helfen, Trends zu erkennen, Profile zu schärfen oder Suchen zu optimieren. Dieser Artikel erläutert die Grundlagen der Text-Analyse und gibt einen Einblick in Volltextsuche, Natural Language Processing und Machine-Learning-Verfahren.

Unstrukturierte Daten sind Daten, die für konventionelle Verarbeitungssysteme keine erkennbare Struktur aufweisen. Der Anteil dieser Daten an allen digital gespeicherten Informationen beträgt etwa 80 Prozent. Bislang beschränkt sich die Mehrheit aller Werkzeuge zur beschreibenden und vorhersagenden Analyse auf die verbleibenden 20 Prozent. Dabei findet man unstrukturierte Daten überall. Sie lassen sich grob in zwei Gruppen untergliedern: in maschinengenerierte und in von Menschen generierte Daten.

Zu ersteren gehören vor allem Bilder und Videos, die von einer Kamera erzeugt werden, aber auch Satelliten-, Radar- und Sonaraufnahmen. Die von Menschen generierten unstrukturierten Daten besitzen meist eine übergeordnete gemeinsame Struktur: die Syntax der Sprache, die diese Menschen sprechen. Zu diesen Daten gehören E-Mails, Dokumente, Social-Media-Daten, SMS und andere Textnachrichten sowie der Inhalt von Internetseiten.

```
Oracle Text:
CREATE INDEX my_index ON MyDocs(my_text)
   INDEXTYPE IS CTXSYS.CONTEXT PARAMETERS
   ('FILTER CTXSYS.NULL_FILTER SECTION GROUP CTXSYS.HTML_SECTION_GROUP');

tsearch2:
UPDATE MyDocs my_index = to_tsvector('german', coalesce(my_text, ''));

Apache Lucene:
new QueryParser(LUCENE_VERSION, "", new GermanAnalyzer(LUCENE_VERSION)).parse(my_text);
```

Listing 1: Funktionen der Text-Extraktion in verschiedenen Textanalyse-Frameworks

Diese Formate enthalten zwar nicht die Informationsdichte strukturierter Formate, doch allein aufgrund der ungeheuren Menge an verfügbaren unstrukturierten Daten darf diese Informationsquelle bei ganzheitlichen Analyseansätzen nicht ignoriert werden.

Klar abzugrenzen von dieser Definition sind demnach Datenformate, die eine klare, wenn auch nicht schematisch festgelegte Struktur aufweisen. Darunter fallen beispielsweise Sensordaten, Serverlogs, XML- und JSON-Dokumente sowie Graphen. Diese Formate werden meist unter den semistrukturierten Daten zusammengefasst.

Text-Extraktion

Unabhängig davon, welches Ziel man bei der Arbeit mit Texten verfolgt, steht normalerweise am Anfang stets die Text-Extraktion, ein Verfahren, mit dem aus dimensionslosen Satzkonstrukten Einzelbausteine des Textes extrahiert und in eine normalisierte Form gebracht werden können. Eine Text-Extraktion läuft in vier Schritten ab:

- *Filtering*
  Das Entfernen nicht benötigter Interpunktion und die Zusammenführung von getrennten Begriffen, etwa beim Zeilenumbruch mit Bindestrich.
- *Tokenization*
  Unterteilung des Textes in einzelne Wörter. Hierbei werden Bindestriche und andere sprachabhängige Gegebenheiten zur Verkettung von Wörtern berücksichtigt.
- *Recognition*
  Die Erkennung von Entitäten unter den Tokens. Je nach Text-Parser kommen hier unterschiedliche Sets an Wörterbüchern und reguläre Ausdrücke („RegEx“) zum Einsatz. So werden häufig Namen, Orte, Geldbeträge, Jahreszahlen, Daten und Telefonnummern erkannt. Auch sogenannte „Stop Words“, also Wörter, die für die spätere Weiterverarbeitung aufgrund ihrer Häufigkeit und fehlenden Aussagekraft unnötig sind (wie „ist“, „werden“, „machen“), werden in diesem Schritt entfernt.
- *Stemming*
  Die Reduzierung aller Wörter, die im Schritt „Recognition“ nicht als Eigennamen identifiziert worden sind, auf ihren Wortstamm. So wird aus „schreiben“, „schrieb“, „schreibt“ oder „geschrieben“ in allen Fällen „schreib“. Auf diese Art und Weise entfällt beim späteren Wortvergleich oder bei Suchfunktionen die Komplexität, die eine Sprache durch Deklination und Konjugation bekommt.

Dieses zunächst äußerst kompliziert wirkende Verfahren wird heute von allen Werkzeugen zur Textanalyse und Volltextsuche unterstützt und lässt sich üblicherweise mit einem einzigen Befehl ausführen. *Listing 1* zeigt dies an drei Beispielen.

Diese vorhandenen Werkzeuge eignen sich natürlich nur dann für die Anwendung, wenn die zur Verfügung stehenden Text-Extraktoren – also Recognition-Wörterbücher, RegEx für „Stop Words“ und weitere zu erkennende Entitäten – ausreichen. Ist dies nicht der Fall, müssen für den Extraktionsprozess eigene Text-Extraktoren entwickelt werden.

Unabhängig von der verwendeten Technologie verläuft die Entwicklung der Text-Extraktoren nach einem ähnlichen Muster. Als Grundlage für die Annotation des Textes wird ein repräsentativer Beispielauszug des Gesamtdatenbestands in das Entwicklungssystem importiert. Der Entwickler definiert die Attribute und Formate, die er als Ergebnis der Analyse erhalten möchte, und nutzt Wörterbücher sowie RegEx, um alle Wörter in den Beispieltexten zu annotieren, die er für die Ermittlung seiner Zielattribute benötigt. Diesen Vorgang wieder-

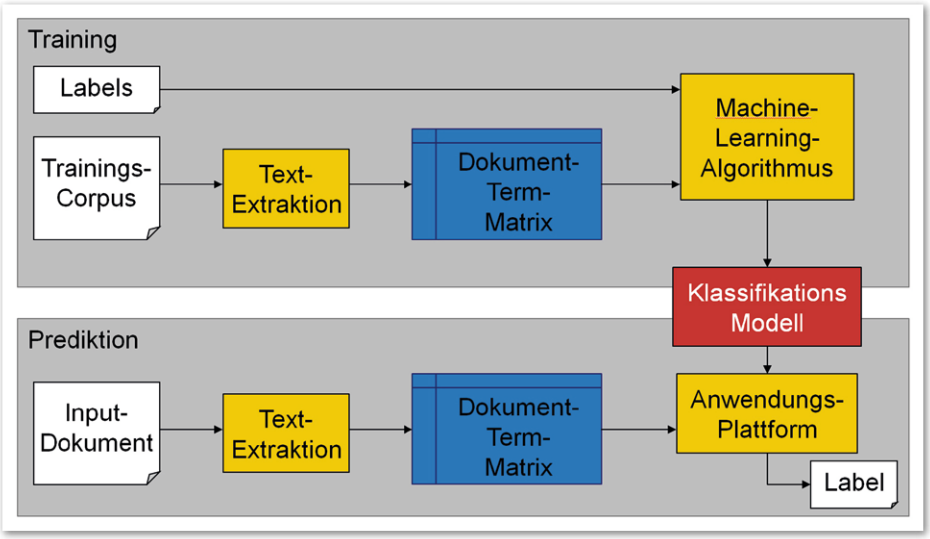

Abbildung 1: Evolution der Textanalyse

holt er inkrementell, um eine wachsende Abdeckung zu erzielen.

Natural Language Processing

Alternativ zu Wörterbüchern und regulären Ausdrücken bieten Natural Language Processing Engines (NLP) weitere Verfahren, um die extrahierten Tokens eines Textes mit weiteren Attributen zu annotieren. Dies können beispielsweise sprachsyntaktische Informationen der Grammatik sein. Das Angebot an NLP-Engines ist vielfältig – ob Open Source oder kommerziell verfügbar. Es gibt hier keine klaren Marktführer, da die Anforderungen sehr unterschiedlich sind und die Analyse-Qualität je nach Anwendungsfall stark variieren kann. Zu den grundlegenden Funktionalitäten einer NLP-Engine gehören:

- Satzerkennung, nicht nur anhand von Interpunktion, sondern auch anhand des Vorhandenseins aller Satzbestandteile
- vorgefertigte RegEx für die Erkennung von Geldbeträgen, Namen und Ähnlichem
- Categorizer, die meist basierend auf einem sehr generischen Corpus den Text einer Gattung zuordnen
- Syntax Parser, die alle Tokens eines Satzes in ihren syntaktischen Eigenschaften bestimmen

Mithilfe dieser Anreicherungen können weitere Filterungen durchgeführt oder Bezüge hergestellt werden. So kann zum Beispiel bei einer Suche explizit zwischen dem substantivierten Gebrauch des Worts und dem Gebrauch als Adjektiv unterschieden werden.

Volltextsuche

Volltextsuche ist weit mehr als die Suche einer Zeichenkette in einem Text oder die Anwendung eines regulären Ausdrucks. Sie bildet die Grundlage für höherentwickelte Textanalyse-Verfahren *(siehe Abbildung 1)*:

- *Lexem-Erkennung*
  Mittels Stemming werden Worte unabhängig von ihrer Deklination oder Konjugation gefunden
- *Volltext-Indizierung*
  Durch die Indizierung der Texte ist die Suche deutlich schneller möglich als mit aktuellen RegEx-Interpretern
- *Ranking-Informationen*
  Informationen, die zur Bestimmung des Grads der Übereinstimmung eines Textes mit dem Suchbegriff herangezogen werden können
- *Interpretation von Satzbau und Satzzeichen via NLP*
  Beispielsweise würde eine Volltextsuche nach „neue Informationen schnell gewinnen“ in einem Text auch die Passage „durch schnelle Informationsgewinnung“ finden, was mit dem bloßen Matching von Zeichenketten oder mit RegEx nicht möglich gewesen wäre.

Die Volltextsuche gliedert sich in die Indizierung der Texte und die eigentliche Abfrage. Bei der Indizierung wird nach der Text-Extraktion ein Index für alle Tokens in allen indizierten Texten angelegt. Dieser Index beinhaltet das normalisierte Token – also das normalerweise auf den Wortstamm reduzierte Wort – sowie alle Positionen im Text, an denen das Token auftaucht. Bei der Suche wird der eigentliche Text also gar nicht prozessiert, sondern es wird lediglich der Volltext-Index mit den Suchbegriffen abgeglichen, die zuvor den Prozess der Text-Extraktion durchlaufen haben.

Neben dem schnellen Finden von Übereinstimmungen in Texten ist das Ranking der Suchergebnisse ein wichtiger Bestandteil der Volltextsuche, um diese dem Nutzer beziehungsweise der verarbeitenden Applikation in nach Relevanz sortierter Reihenfolge zur Verfügung zu stellen. Die dabei für das Ranking herangezogenen Faktoren können je nach Anwendungsfall variieren, üblicherweise fließen jedoch mindestens diese drei Faktoren mit ein:

- *Lexikalische Relevanz („lexical")*
  Wie häufig kommen Suchbestandteile in dem Dokument vor beziehungsweise wie hoch ist die Trefferdichte?
- *Nachbarschaftsrelevanz („proximity")*
  Wie nahe liegen die Suchbestandteile (bei der Suche mehrerer Begriffe) beieinander?
- *Strukturelle Relevanz („structural")*
  Wie wichtig sind die Teile des Textes, in denen die Suchbegriffe vorkommen?

Diese Punkte werden oft mit weiteren Meta-Informationen wie Aktualität des Textes, Bezügen von anderen Dokumenten, Format, Textgattung etc. kombiniert. Jeder dieser Faktoren wird dann üblicherweise als ein normalisierter Wert zwischen 0 und 1 abgebildet und mit den anderen Faktoren multipliziert. Das Ergebnis liefert eine gute erste Einordnung, für eine Feinabstimmung muss jedoch noch an den oft manuell zu setzenden Wichtungsfaktoren gedreht werden.

Zum Beispiel kann die lexikalische Relevanz, also die Dichte der Suchbegriffe in den Texten, auf diverse Arten errechnet werden: Die einfachste Methode ist eine einfache Division, sodass bei einem Treffer die Relevanz des Dokuments mit zunehmender Länge linear abnimmt. Bessere Ergebnisse liefert das „tf-idf"-Maß, das statt der Länge des Dokuments die Häufigkeit des am häufigsten genutzten Worts als Divisor verwendet. Oft werden diese Ergebnisse für eine feinere Abstufung noch logarithmiert, durch Hauptachsen-Transformationen mit einem Schwellwert versehen oder progressiver beziehungsweise degressiver gemacht.

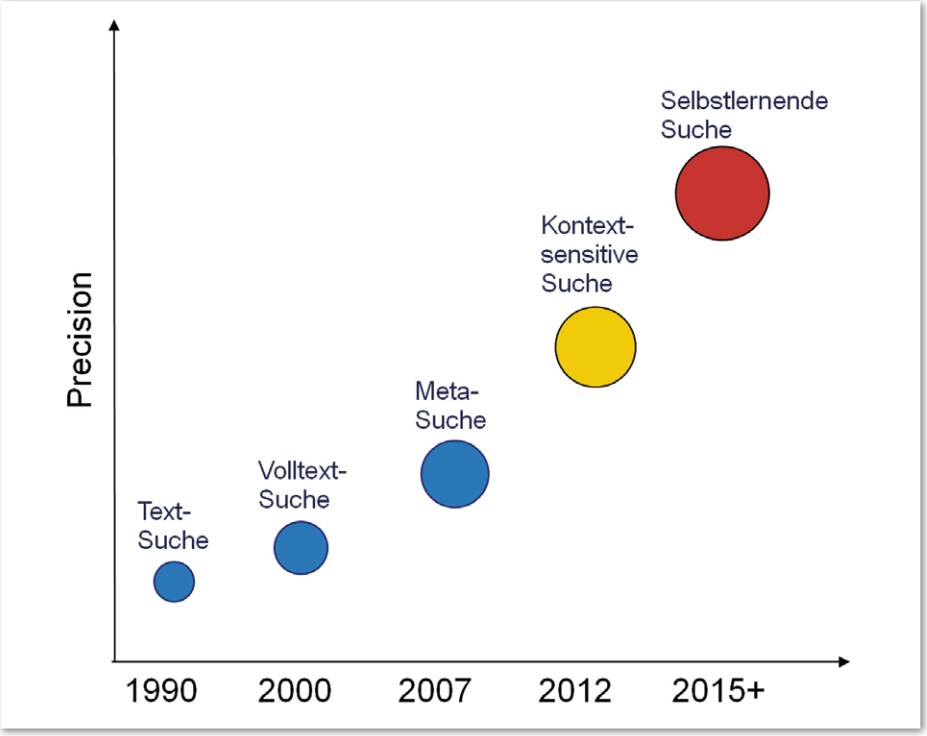

Abbildung 2: Training und Anwendung eines Klassifikationsmodells durch überwachtes Lernen

Text-Klassifikation

Gerade im Bereich der Textsuche und -analyse ergibt sich eine Vielzahl von Faktoren, die für eine Texteinordnung, -suche oder -bewertung eine Rolle spielen können, aber stets nur in Kombination mit anderen Faktoren ein nutzbares Resultat liefern. Die Abstimmung dieser Faktoren aufeinander spielt hierbei eine entscheidende Rolle und kann häufig nur bis zu einem bestimmten Grad manuell vorgenommen werden.

Die Klassifizierung von Texten ist einer der häufigsten Anwendungsfälle für die Textanalyse. Hierbei soll ein unbekannter Text einer oder mehreren vorher definierten Gruppen von Texten zugeordnet werden. Die Ergebnisse sehen wir täglich, etwa wenn Google Werbung anzeigt, die zum einen auf uns persönlich angepasst ist, zum anderen aber auch auf dem Inhalt der Seite basiert, die wir gerade ansehen. Oder wenn unser Google-Gmail-Konto uns automatisch vorschlägt, in welche Unterordner es eingehende Mails aufgrund ihres Inhalts einsortieren würde. In Support-Centern werden Classifier häufig verwendet, um Mails nach ihrer Thematik vorzusortieren, damit sie schneller bei der richtigen Person landen. Und im Bereich der Textsuche werden Classifier verwendet, um für eine Suche zu bestimmen, in welchem Kontext die Suche überhaupt stattfinden soll.

Suchen wir beispielsweise bei Google nach „Apache Tomcat", finden wir seitenweise Treffer zum gleichnamigen Applikationsserver. Wir finden keine Ergebnisse aus dem Bereich des U.S. Militärs, dessen Luftstreitkräfte sowohl über das Grummann F-14 Tomcat Kampfflugzeug als auch den Boeing AH-65 Apache Kampfhubschrauber verfügen. Google schließt aus der Kombination dieser beiden Begriffe, dass der Kontext „IT" und nicht „Militär" ist, und gewichtet Resultate höher, die der Kontextgruppe „IT" zugeordnet sind.

Ein Classifier ist also ein Bemessungssystem, in dem eine Entität anhand ihrer Attribute ein oder mehrere Labels erhält. Er kann in einfachster Form durch ein regelbasiertes System manuell abgebildet

werden, in dem hinter jeder Ausprägung Regeln hinterlegt sind.

Solche Systeme werden meist mithilfe von Business Rule Engines wie Apache Drools abgebildet. Begibt man sich in den Bereich der maschinengestützten Lernverfahren, das sogenannte „Machine Learning", unterscheidet man grob zwei Lernformen: überwachtes und unüberwachtes Lernen. In beiden Fällen wird ein Set von Texten benötigt, mit denen das System lernen kann. Dieses als „Corpus" bezeichnete Trainings-Datenset durchläuft den Prozess der Textextraktion und wird dann in eine Dokument-Term-Matrix zerlegt, also eine Tabelle, die die Häufigkeit einzelner Wörter im selben Text aufzeigt. Diese Matrix kann nach Bedarf transformiert werden, um etwa für jedes Wort zu bestimmen, wie oft es mit jedem anderen Wort im Schnitt im gleichen Text vorkommt.

Diese Matrizen bilden also die Grundlage für den zu trainierenden Classifier. *Abbildung 2* stellt den Trainings- und Ausführungsprozess vereinfacht dar. Überwachtes und unüberwachtes Lernen unterscheiden sich hier lediglich in dem Punkt, dass beim überwachten Lernen die Labels, also die Gruppen in die klassifiziert werden soll, vorgegeben sind, während beim unüberwachten Lernen häufig nur die Anzahl der Gruppen, die gebildet werden sollen, vorgegeben wird und der Algorithmus die Definition der Gruppen selbst optimiert. Das exakte Verfahren zur Erstellung des Classifiers aus dem Corpus hängt vom verwendeten Lernalgorithmus ab.

Bekannte Klassifikationsverfahren des überwachten Lernens sind der Bayes-Klassifikator, das Perzeptron (neuronales Netz) und die Support Vector Machine (SVM). Verfahren des unüberwachten Lernens sind zumeist Segmentierung/Clustering-Verfahren.

Text Mining oder Volltextsuche

Volltextsuche ist mittlerweile ein gestandenes Verfahren mit etablierten Technologien, während gerade mit dem Big-Data-Hype immer neue Data- und Text-Mining-Werkzeuge für skalierbare Plattformen wie „Spark on Mesos" oder „Hadoop" auftauchen. Beide Verfahren ähneln sich in der Aufbereitung der Daten, ihre Funktion ist jedoch auf ihren primären Anwendungszweck ausgelegt. Bei der Volltextsuche heißt dies, für einen Suchbegriff möglichst rasch eine nach Relevanz bewertete und sortierte Liste mit Antworten zu liefern, während die Textanalyse beziehungsweise das Text Mining das Ziel verfolgt, einen Zusammenhang zwischen den Texten herzustellen, und daher meist mit Machine-Learning-Verfahren verknüpft ist.

In der Praxis gehen beide Ansätze allerdings meist Hand in Hand. Nehmen wir zum Beispiel eine Textsuchmaschine, so ist die Definition von Synonymen und Ähnlichkeiten zwischen Begriffen ein wichtiger Aspekt. Diese Synonyme und Wortbeziehungen können je nach Kontext unterschiedlich sein. Während das englische Wort „Spring" (Frühling) in einem generischen Kontext nichts mit dem Begriff „Struts" (Säulen) zu tun hat, haben diese beiden Begriffe im IT-Kontext durch einst sehr verbreitete Java-Frameworks eine starke semantische Beziehung. Die intelligente Suchmaschine eines IT-Unternehmens sollte bei einer Suche nach „Java" also auch Texte finden, in denen zwar das Wort „Java" nicht vorkommt, jedoch Begriffe wie „Servlet", „Spring" oder „Hibernate".

An dieser Stelle kommen Text-Mining-Verfahren ins Spiel, die auf Basis eines großen Text-Corpus aus dieser IT-Firma bereits die Beziehungen zwischen allen Begriffen erlernen und in einer indizierten Synonymtabelle mit entsprechenden Korrelationsfaktoren ablegen. Unabhängig von der Suche erstellt ein solches Verfahren in selten ausgeführten Batchläufen, die aufgrund ihrer Rechenanforderung oft viele Stunden oder Tage dauern können, die für die Suche zur Laufzeit benötigten indizierten Informationen, um Suchanfragen schnell und präzise beantworten zu können.

Ein Wort zur Infrastruktur

Aufgrund ihres sehr unterschiedlichen Profils laufen Anwendungen für Text Mining und Textsuche meist auch auf technologisch getrennten Plattformen. Mit dem Big-Data-Hype werden für das Text Mining skalierbare, offene Plattformen wie Hadoop und Spark immer populärer, besonders wenn die Durchlaufzeit für die Batchläufe zeitkritisch ist.

Die Textsuche wird meist in relationalen Datenbanken wie Oracle Text in der Oracle-Datenbank oder tsearch2 in der PostgreSQL-Datenbank ausgeführt, dort sind diese ein bewährter Bestandteil. Die zuvor erstellten Modelle werden hierzu in ein relationales, tabellarisches Schema übertragen. Speziell dafür ausgelegte Text-Index-NoSQL-Datenbanken wie Solr oder ElasticSearch werben hier mit deutlich besserer Skalierbarkeit zu geringeren Kosten, besitzen jedoch nicht die Allrounder-Funktionalitäten einer relationalen Datenbank. Hier ist abzuwägen, ob die erwartete Indexgröße die Leistungsfähigkeit eines einzelnen Datenbanksystems übersteigen würde, was aber meist erst bei einem Index über viele Millionen Dokumente der Fall ist.

Bei all dem Wirbel um die Arbeit mit unstrukturierten Daten und den vielen neuen Produkten in diesem Umfeld sollte man sich immer vor Augen führen, dass es sich hier keinesfalls um ein neues Thema handelt. Letzten Endes läuft weiterhin alles auf einen Daten-Parser hinaus, der entsprechend seinem Reifegrad bei der Textverarbeitung viele Dinge von selbst übernimmt, etwa mathematische Verfahren zur Hauptkomponenten-Analyse und Clustering sowie auf an textuelle Datenformate angepasste Indizes; alles Verfahren, für die uns seit Ende der 1990er-Jahre bereits Werkzeuge zur Verfügung stehen, die derzeit allerdings unter dem Hype-Begriff „Big Data" einen zweiten Frühling erleben.

Christopher Thomsen
christopher.thomsen@opitz-consulting.com