DBPRO: Realisierung eines Systems für Indizierung, Clustering und Suche von Dokumenten

Franz Tscharf, Henrik Fröhls und Eric Schneider

Projektbericht Datenbankprojekt
Fakultät IV
Elektrotechnik und Informatik
der Technischen Universität Berlin

Betreuer:
Holmer Hemsen

Fachgebiet:
Datenbanksysteme und Informationsmanagement

Berlin den 13. März 2018
Wintersemester 17/18

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abstrakt

Der vorliegende Abschlussbericht des Moduls „Datenbankprojekt“ im Wintersemester 2017/2018 in Zusammenarbeit mit dem Fachgebiet Datenbanksysteme und Informationsmanagement an der Technischen Universität Berlin beschreibt das Semesterprojekt „Dokumenten Clustering“. Ziel des Projekts ist ein System für die Indizierung, Clustering und Suche von Dokumenten zu entwickeln. Es wurde ein Java Spring Web-Server mit Bootstrap, jQuery und Foamtree entwickelt, der die Daten mit Kibana visualisiert und mit Elasticsearch durchsuchbar macht. Das Clustering der Dokumente auf Basis der Metadaten wurde mit Carrot^2 realisiert.

1. Einleitung

Das im Folgenden beschriebene Projekt „Dokumenten Clustering“ wurde im Rahmen des Moduls „Datenbankprojekt / Informationsmanagementprojekt“ durchgeführt.
Das Projekt wurde durch Dr. Holmer Hemsen, Wissenschaftlicher Mitarbeiter (Intelligente Analytik für Massendaten) betreut.

Im heutigen digitalen Zeitalter wächst das Datenvolumen rasant an und eine Trendwende ist nicht absehbar. Um diese Datenmenge verarbeiten zu können ist eine Suchfunktion unabdinglich und eine immer zentraler werdende Komponente in unserem Alltag. Eine übersichtliche Darstellung der Ergebnisse sowie der Zusammenhänge zwischen diesen ist notwendig, um sinnvoll mit ihnen umgehen zu können.
Clusteringalgortihmen können ein wertvolles Werkzeug sein um einen Überblick über Dokumente und deren Inhalt zu bekommen.
Eine ansprechende grafische Darstellung solcher Cluster auf einer benutzerfreundlichen Oberfläche umzusetzen ist ein fundamentaler Teil dieses Projekts. Hinzu kommt die Visualisierung von Metadaten in verschiedenen Darstellungen.
In den folgenden Kapiteln wird auf die unterschiedlichen Dienste eingegangen, welche in der Umsetzung Anwendung gefunden haben. Außerdem werden die eingesetzten Methodiken und Vorgehensweisen skizziert, sowie die Realisierung im Detail beschrieben.

2. Vorgehen

2.1 Zielsetzung

Die Ziele der Anwendung wurden in Meilensteine definiert. Sie wurden am Ende einer jeden Projektphase festgelegt. Auf die wichtigsten Meilensteine des Projektablaufs wird in Tabelle 2.1 näher eingegangen.

Nr.	Datum	Titel	Inhalt
1	16.10.17	Einführung	Organisatorisches, Vorstellung der Projektvorschläge
2	23.10.17	Projektauswahl	Gruppenbildung, Zuteilung der Projekte
3	13.11.17	Meilenstein 1	Management, Zielsetzung, Elaboration der Tools, Erstellung einer Literatursammlung
4	11.12.17	Meilenstein 2	Installation/Konfiguration/Orchestrierung der Dienste, Entwicklung eines Prototypen
5	22.01.18	Meilenstein 3	Web-Service- und Frontend-Entwicklung, Server-Deployment, Konvertierungstool
6	05.02.18	Finale Abgabe	Abgabe des finalen Abschlussberichts
7	12.02.18	Finale PP	Fazit, Ausblick, Persönlicher Lernerfolg

Tabelle 2.1: Meilensteine

Zusammengetragene wissenschaftliche Artikel sollten mittels Elasticsearch indiziert werden und durch das Clusteringframework Carrot[2] zur weiteren Suche aufbereitet werden. Kibana konnte genutzt werden um Daten darzustellen. Am Ende des Semesterprojektes sollte eine ansprechende Web-GUI die Suche sowie Visualisierung ermöglichen.

2.2 Projektmanagement

Im folgenden Abschnitt sollen die Methoden und Vorgehensweisen des Projektmanagements beschrieben werden, welche auf Basis der ersten Meilenstein-Präsentation festgelegt wurden.

2.2.1 Vorgehensmodell, Methoden und Hilfsmittel

Das Vorgehensmodell bildet die Basis aller weiteren Aktivitäten des Projektmanagements. Durch die Entscheidung für ein agiles Vorgehen, konnten wir unsere Arbeitschritte weitestgehend separat betrachten und auf die zuständigen aufteilen. In unseren Wöchentlichen Treffen haben wir den aktuellen Stand besprochen und weiter Schritte Diskutiert. Um das Projekt zu planen, zu verwalten und die „Issues“ zu Dokumentieren haben wir den Dienst waffel.io verwendet, der exzellent mit Github interagiert. Durch waffle.io[1] konnten wir die Probleme skizzieren und Verantwortlichen zuordnen.

1. *Developer-first Project Management for Teams on GitHub*, besucht am 25. Januar 2018, `https://waffle.io`.

Die Kommunikation im Projektteam bildete die Grundlage für ein erfolgreiches Wissensmanagement, die Verteilung von Aufgaben und der Organisation von Arbeitstreffen. Die Herausforderung in der Kommunikation bestand insbesondere darin, die verschiedenen Kommunikationsanlässe so zu strukturieren, dass alle benötigten Informationen zum richtigen Zeitpunkt den richtigen Teammitgliedern zur Verfügung standen und dabei eine gewisse Übersicht gewahrt werden konnte. Die Software Slack konnte genau diese Anforderungen erfüllen. Hier erstellten wir folgende Untergruppen (Abb. 2.1), um unsere Kommunikation besser zu strukturieren.

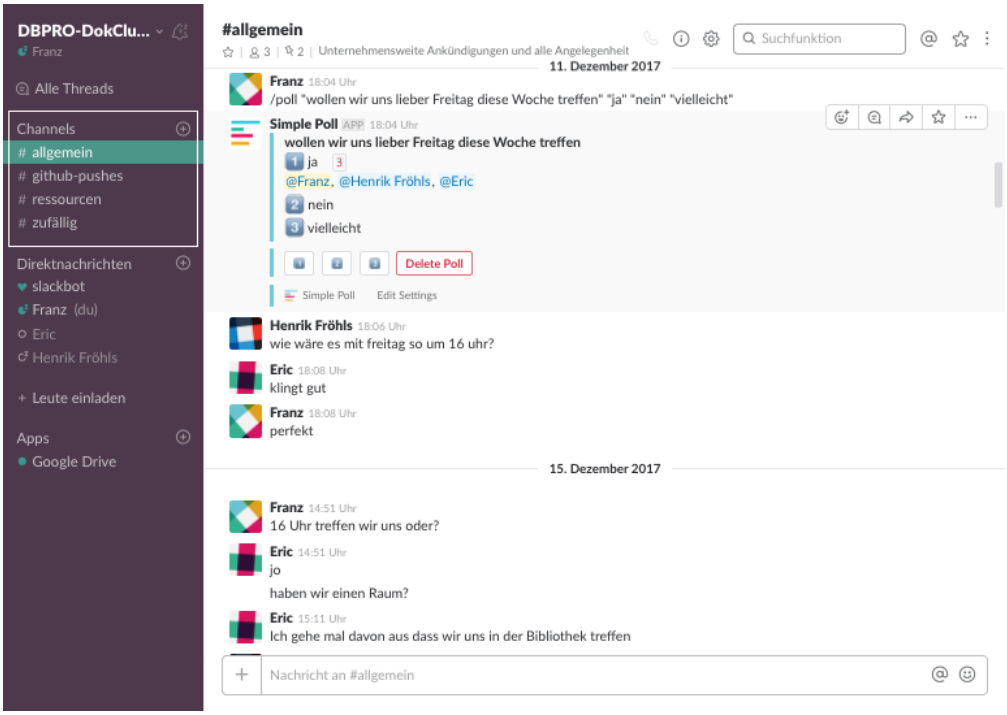


Abbildung 2.1: Slack Channels

2.3 Beschreibung der Realisierung

Nachdem das Projektziel erläutert wurde, soll nachfolgend die Realisierung beschrieben werden, mithilfe dessen die Anforderungen gelöst werden sollten.

2.3.1 Zotero

Zotero ist ein kostenloses, einfach zu bedienendes Werkzeug zum wissenschaftlichen Arbeiten. Es hilft dabei, Quellen zu sammeln, zu organisieren, zu zitieren und zu teilen. Gerade beim Verfassen von Seminar- und Abschlussarbeiten hilft Zotero, um bibliographische Informationen zu sammeln und diese effizient einzusetzen.
In dieser Arbeit wurde Zotero verwendet um die Metadaten von Dokumenten einer vorgegebenen Domäne (‘Signal Processing’) zu sammeln. Für das Literaturmanagementsystem sollte zusätzlich in den Einstellungen die Option der PDF-Indizierung gewählt werden. Durch diese Auswahl kann Zotero auch PDF Dokumenten analysieren und durchsuchen. Um die Zotero Dokumentenbibliothek von Elasticsearch zu

indexieren muss anschließend in der Zotero Connector GUI die Bibliothek als CSL JSON exportiert werden. Dafür wählt man Datei → „Bibliothek exportieren...“ → Format: CSL JSON. Im nächsten Schritt muss die Datei zu einem data binary JSON Format konvertiert werden. Dafür wird das von uns entwickelte Java Anwendung, das in der Git Repository zu finden ist. Um die Datei anschließend durch ein bulk post Request an die Elasticsearch API zu senden, kann das `./doc/setup.sh` Skript ausgeführt werden.

2.3.2 Verwendung der JSON-Umwandlung Anwendung

Die Umwandlung der Zotero CSL JSON Datei ist möglich über die Java-Json-to-ES-Bulk.jar, welche sich in dem Ordner `applications/Java-Json-to-ES-Bulk/out/artifacts/Java_Json_to_ES_bulk.jar` befindet und extra für dieses Projekt entwickelt wurde. Diese Datei nimmt als Eingabeparameter einen absoluten oder relativen Pfad zu einer CSL JSON Datei von Zotero an und kreiert im gleichen Ordner folgende Datei `request-data.json`. Diese Datei kann nun verwendet werden um Daten zu indizieren (Beschrieben in Kapitel 2.3.1). Innerhalb des Java-Json-to-ES-Bulk Ordners befindet sich auch der Quellcode der Anwendung. Die Funktionsweise dieser Anwendung ist recht einfach. Es iteriert durch das Array an Dokumenten, welche sich in der Zotero CSL JSON Datei befinden und wandelt alle Daten Felder von

```
"issued":{
  "date-parts": [
      "2017":
        9
    ]
}
```

in das ISO 8601 Datumsformat:[2] `"issued": "2017-09"` zusätzlich werden die Autoren Felder von

```
"author": [
  {
     "family": "Djigan",
     "given": "V. I."
  }
]
```

in folgendes Format umgewandelt: `"author":["Djigan, V. I."]`, also in ein Array in dem Format Familienname gefolgt von einem Komma und dann gefolgt vom Vornamen. Zusätzlich werden die Einträge in das Bulk[3] Format von Elasticsearch gebracht, welches wie folgt aussieht:

```
{"index":{"_index":"zotero","_type":"entry","_id":1}}
{"container-title":"2017 IEEE East-West Design Test Symposium (EWDTS)","
    author":["Djigan, V. I.","Shakhtarin, B. I.","Likhoedenko, K. P.","
    Zhurakovskiy, V. N."], ...}
{"index":{"_index":"zotero","_type":"entry","_id":2}}
```

2. Misha Wolf und Charles Wicksteed, *Date and Time Formats*, besucht am 27. Januar 2018, `https://www.w3.org/TR/NOTE-datetime`.

3. *Bulk API — Elasticsearch Reference [6.1] — Elastic*, Learn/Docs/Elasticsearch/Reference/6.1, besucht am 28. Januar 2018, `https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html`.

```
{"note":"DOI: 10.1016\/B978-0-12-811534-3.00002-0","author":["Lei, Yaguo
    "],"source":"ScienceDirect","abstract":"This chapter mainly ...}
...
```

Das Format besteht zunächst aus einer Zeile mit Informationen über ein mitgegebenes Objekt sowie die darauf auszuführende Aktion, gefolgt von dem Objekt. Ein Beispiel Aufruf mit der Zotero JSON Datei im selben Ordner ist dann

```
java -jar Java_Json_to_ES_bulk.jar SignalProcessing.jar
```

2.3.3 Elasticsearch

Elasticsearch kann beschrieben werden als eine Open-Source-Suchmaschine. Die Software wurde für den Einsatz auf verteilten Systemen und die Echtzeit-Verarbeitung von großen Datenmengen entwickelt. Vorwiegend wird das Tool zur Volltextsuche als auch zur Suche in strukturierten Daten und zu Analysezwecken eingesetzt. Elasticsearch ist ein in Java geschriebener Suchserver, der JSON-Dokumente speichern und durchsuchen kann. Mit dem Elasticsearch-Server kann über eine RESTful API kommuniziert werden. Diese Art von Kommunikation bietet einige Vorteile, darunter Skalierbarkeit, Komposition von Diensten und Anbindung von Fremdsystemen. Die Kompatibilität zu Fremdsystemen spielt besonders im vorliegenden Projekt eine wichtige Rolle.

2.3.4 Kibana

Kibana ist ein browserbasierte Analyse- und Such-Software und wird hauptsächlich in Kombination mit Elasticsearch verwendet. Für unerfahrene Anwender visualisiert Kibana die Daten in Form eines Dashboards und bietet gleichzeitig für erfahrene Anwender eine mächtige Analyse- und Suchfunktion. Die Kibana-Oberfläche bietet diverse Komponenten um aus den Datensätzen die einzelnen Diagramme zu erstellen.
Im vorliegenden Projekt wird die Kibana-Oberfläche verwendet um die in Elasticsearch indizierten Dokumente zu visualisieren und zu analysieren. Abfrageorientiert ist eine Darstellung der Inhalte möglich.

2.3.5 Kommunikationsarchitektur

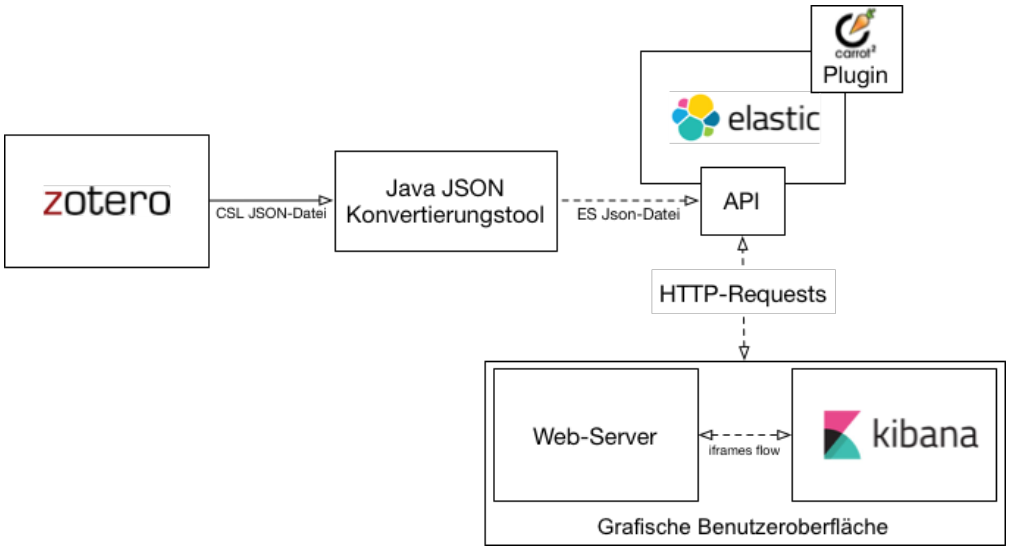


Abbildung 2.2: Die Kommunikationsarchitektur der Dienste

3. Konfiguration der Server

Im Laufe des Projekt wurden die Server Kibana und Elasticsearch aufgesetzt und konfiguriert, um mit dem entwickelten Web-Service zu kommunizieren. In dem folgenden Abschnitt wird eine kurze Einführung in die Konfiguration und Installation der verschiedenen Dienste gegeben.

3.1 Elasticsearch

Bevor Elasticsearch installiert wird, sollte OpenJDK auf dem System zur Verfügung stehen. Die Elasticsearch Version 5.5.2 kann auf der folgenden Website heruntergeladen werden. https://www.elastic.co/downloads/past-releases/elasticsearch-5-5-2
An dieser Stelle ist wichtig zu erwähnen, dass für die Anbindung von Carrot[2] ein Plugin für Elasticsearch verwendet wurde. **Das Plugin unterstützt ES nur bis Version 5.5.2.** Bitte Achten Sie auf das Verwenden einer kompatiblen Version. Für die Installation des Plugins sind folgende Schritte notwendig:

1. Die Dokumentation ist zu finden unter https://github.com/carrot2/elasticsearch-carrot2
2. Herunterladen der Erweiterung durch den ES Plugin installer mit dem Befehl:
   `./bin/elasticsearch-plugin install org.carrot2:elasticsearch-carrot2:6.1.1`
3. Um das Plugin mit der Elasticsearch API zu nutzen müssen die CORS requests erlaubt werden. Bearbeiten Sie dafür die ES/config/elasticsearch.yml Datei und fügen Sie die folgende Zeilen hinzu.

```
# Allow localhost cross-origin requests.
http.cors.enabled: true
http.cors.allow-origin: /(null)|(https?:\/\/localhost(:[0-9]+)?)|(
    https?:\/\/cdn\.rawgit\.com(:[0-9]+)?)/
```

Um ES für Testzwecke auch im lokalen Netzwerk ansprechen zu können muss in der `./config/elasticsearch.yml` Datei der `network.host: 192.168.1.215` und `discovery.type: single-node` gesetzt werden. Die IP im obigen Beispiel entspricht unserer Testumgebung muss, aber in anderen Konfiguration angepasst werden. Abschließend kann durch den Befehl `./bin/elasticsearch` Elasticsearch ausgeführt werden. Um die Beispieldatenbank in Elasticsearch zu importieren kann `./doc/setup.sh` genutzt werden.

3.2 Kibana

Herunterladen der Kibana Version 5.5.2 unter der Website: kibana v. 5.5.2[1] Starten von Kibana über das Terminal durch `./kibana/bin/kibana` Die Benutzeroberfläche von Kibana kann über den Port:5601 durch den Web Client aufgerufen werden. Bei dem ersten Aufruf des Server muss der „index pattern“ eingestellt werden. Der Index unserer Elasticsearch Requests ist `zotero*`. Die Option für das „Time Filter field name“

1. *Kibana 5.5.2*, besucht am 26. Januar 2018, `https://www.elastic.co/downloads/past-releases/kibana-5-5-2`.

wird in unserer Konfiguration nicht verwendet. Im nächsten Schritt müssen die Visualisierungen in Kibana importiert werden. Dafür wählt man *Management* → *Saved Objects* → *Import* Button und importieren die JSON Datei, welche in unserem GitHub Repository unter `./doc/ExportVisualisations.json` zu finden ist.

3.3 Java Spring Web-Server

Der Web-Server, den wir im Laufe des Projekts entwickelt haben, kann über den Befehl `java -jar ../DokClusterWebGUI/target/spring-boot-web-application.war` gestartet werden. Anschließend kann die GUI mit dem Port:8800 über den Web-Client angesprochen werden. Der Quelltext des Servers ist in der Github Repository im Ordner `./DokClusterWebGUI` zu finden. Wir empfehlen den Editor IntelliJ von jetbrains, um den Java Spring Server weiter zu entwickeln.

4. Frontend

4.0.1 Bootstrap

Als Basis für das Frontend wurde das populäre Framework Bootstrap verwendet. Es ermöglicht eine ansprechende übersichtliche Darstellung. Seine Anpassungsfähigkeit und Kompatibilität für responsive Webdesign waren ausschlaggebend bei der Wahl des Frameworks.

4.0.2 Responsive Design

Um die Plattform möglichst vielen Nutzern mit unterschiedlichen (mobilen) Endgeräten zugänglich zu machen, wurde das Web-GUI anpassungsfähig gestaltet. Je nach Browser-Fensterbreite/-höhe werden verschiedene Elemente (u.a. Navigation) versteckt/„eingeklappt“ und Inhalt nebeneinander oder auch untereinander dargestellt.

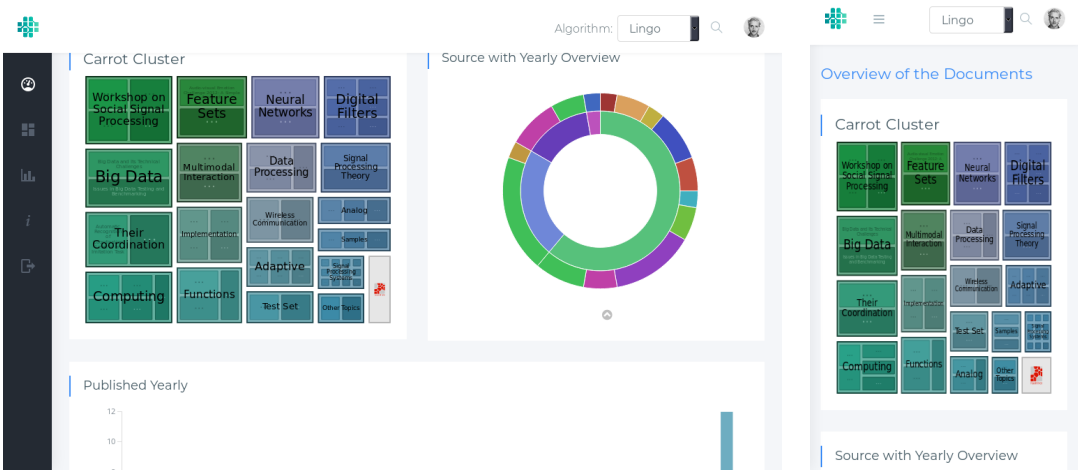


Abbildung 4.1: Dashboard in mobilen Ansichten (Vgl. Abbildung 4.2)

Links: Tabletgröße, linke Navigationsleiste eingeklappt
Rechts: Dashboard in Smartphonegröße, Navigationsleiste über Topbar erreichbar, „Algorithm“ ausgeblendet, Inhalt rearrangiert

4.0.3 Inhalt

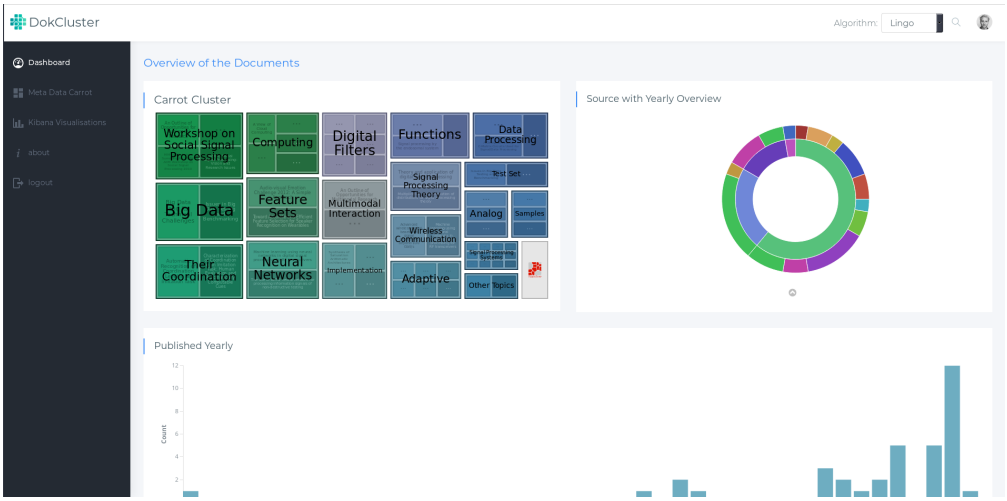


Abbildung 4.2: Dashboard (Full-Sized)

Das Dashboard bildet die „Startseite“ der Web-GUI und bietet einen Überblick über deren Funktionalitäten. An erster Stelle ist die FoamTree Visualisierung des Carrot[2] Clusterings zu sehen. Durch Nutzerinteraktion kann dynamisch durch die Cluster navigiert werden.

Daneben ist ein Kibana Kreisdiagramm der Quellen und Jahre zu sehen. Hier kann der Benutzer durch Anwählen zusätzliche Filter hinzufügen.

Darunter befindet sich ein Kibana Histogramm, welches verdeutlicht, wie viele Artikel pro Jahr veröffentlicht wurden.

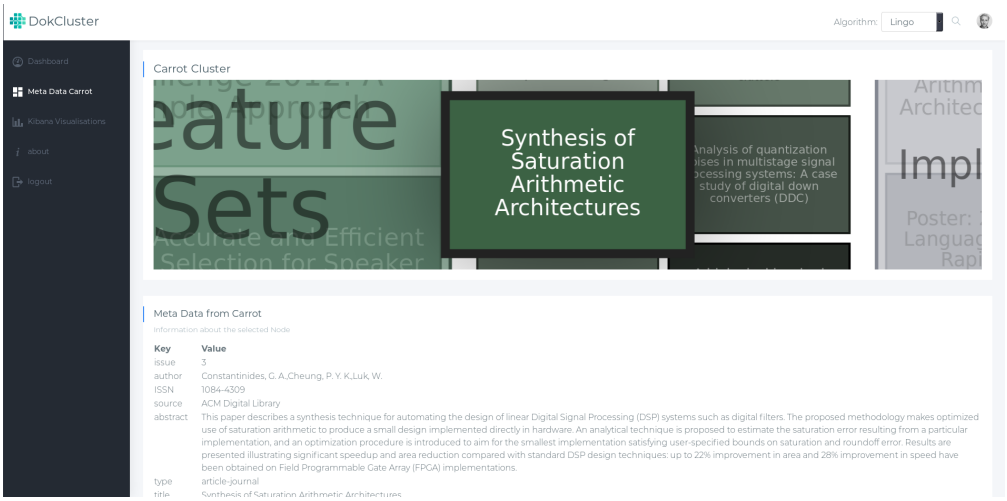


Abbildung 4.3: Meta Data Carrot[2]

Auf der „Meta Data“ Seite kann wie im Dashboard über FoamTree ein Artikel angewählt werden. Zusätzlich werden unter der FoamTree Darstellung alle Meta Daten zum angewählten Artikel angezeigt.

Abbildung 4.4: Kibana Visualisierungen

Im oberen Teil der Kibana Visualisierungen befindet sich ein Histogramm, welches die Arten der Dokumente darstellt. Darunter findet sich eine Heatmap über Quellen und Veröffentlichungsjahre der Artikel. (Detailliertere Darstellung des Pie Charts aus dem Dashboard (Vgl. Abbildung 4.2).)

Zusätzlich zu den gezeigten Seiten umfasst der Webservice noch eine Login-Seite und eine About-Page (Impressum) zur Vorstellung der Gruppenmitglieder und des Projekts.

Wird über die Suchmaske in der Topbar ein neuer Suchbegriff eingegeben, werden auf allen Seiten alle FoamTree- sowie Kibana-Visualisierung automatisch mit dem neuen Parameter neu geladen. In der Topbar kann außerdem in einem Dropdown-Menü der von Carrot[2] anzuwendende Cluster-Algorithmus gewählt werden.

4.0.4 Spring Web MVC

Das Web Framework Spring MVC wurde verwendet, da es gut mit Elasticsearch interagiert. Die Interkompatibilität ist durch die Anwendung von diesem Java-basierten Web-Framework sichergestellt. Mittels Spring Security ist das Anmelden und Abmelden realisiert. So kann eine sichere Authentifikation gewährleistet werden und unberechtigter Zugang zum Web-Service wird verhindert.

4.1 Javascript im Frontend

Vieles im Frontend unseres Projektes ist über Javascript gesteuert. Javascript selbst ist Standard in allen modernen Browsern und ein wichtiger Teil unserer Oberflächengestaltung. Es wird benötigt um bestimmte Ereignisse und Interaktionen mit der Weboberfläche zu steuern. In den folgenden beiden Kapiteln werden die zwei Bibliotheken, welche in unserem Frontend verwendet werden vorgestellt und deren Verwendung erklärt.

4.1.1 jQuery

jQuery[1] ist eine äußerst bekannte und populäre JavaScript Bibliothek, welche eine umfangreiche API und Cross-Browser Unterstützung bereitstellt. Das Frontend nutzt folgende Funktionalitäten:

- das Anvisieren von HTML Elementen durch Selektoren (z.B `$("#element-id")` und `$(".element-class")`)
- die Manipulation dieser DOM Elemente mithilfe der Funktionen `append()` und `empty()`
- das Registrieren von Eventhandlern und
- Aufrufe zu HTTP REST Endpunkten via AJAX

Im folgenden wird nun die Interaktion des Benutzers mit der Oberfläche und dem Frontend beschrieben. Ziel ist es, anhand von konkreten Beispielen, zu zeigen wo jQuery bei diesen Interaktionen des Nutzers eingesetzt wird.
Nachdem sich der Nutzer angemeldet hat und auf die Dashboard Seite weitergeleitet wird, wird direkt nach dem Laden der Seite nach dem Begriff „digital“ gesucht, damit der Nutzer direkt Beispielvisualisierung sieht. Der Begriff „digital“ wurde gewählt, da er gute Ergebnisse, mit den bereits indizierten Daten, erzeugt. Im folgenden werden die beiden Hauptfunktionen erklärt, die die Visualisierungen ermöglichen.
`set_search_graphic` erstellt die FoamTree Visualisierung (siehe Abbildung 4.5) mithilfe des Suchbegriffes und des gewünschten Algorithmus. Zunächst werden die Daten an das Carrot[2] Plugin gesendet via der jQuery Funktion `$.ajax()`. Danach werden die Daten der Antwort weiterverarbeitet und schließlich mit FoamTree visualisiert.
Die andere Funktion `set_kibana_graphic`, die nur den Suchbegriff als Eingabe erhält, aktualisiert die Kibana Grafiken durch Ändern der URL in den `iframe` Elementen, sodass der Suchbegriff enthalten ist. Nach erfolgreichem Laden der Seite, kann der Benutzer nun einen neuen Begriff über das Suchfeld eingeben.
Auf dem `form` Element ist ein Eventhandler registriert welcher beim Nutzen der Entertaste, während das Suchfeld fokussiert ist, `search_textfield_input` erneut aufruft. Auch hier werden die jQuery Selektoren verwendet um den gewählten Algorithmus und das Suchfeld an die Funktion weiterzuleiten. Danach passiert der gleiche Ablauf wie bereits beim Laden der Seite beschrieben wurde, nur dieses Mal mit den Eingaben des Suchfeldes und mit dem gewählten Algorithmus.
Die in diesem Abschnitt beschriebenen Funktionen und Methoden sind in der Datei `DokClusterWebGUI\src\main\resources\static\js\requests\serialize.js` zu finden.
jQuery wird noch an einigen anderen Stellen verwendet, zum Beispiel zum Füllen der Metadatentabelle auf der Metadatenseite oder dem Beginn einer neuen Suchanfrage nach der Wahl eines neuen Algorithmus.

4.1.2 FoamTree

Die FoamTree Bibliothek ermöglicht interaktive Visualisierung von Clustern welche einfach zu bedienen und konfigurieren ist. Der folgende Abschnitt soll erklären wie man die FoamTree Bibliothek nutzt um die Cluster, welche wir vom Elasticsearch

1. jQuery Foundation, *jQuery*, besucht am 25. Januar 2018, `http://jquery.com/`.

Carrot[2] erhalten, zu visualisieren. (Siehe Abbildung 4.5)

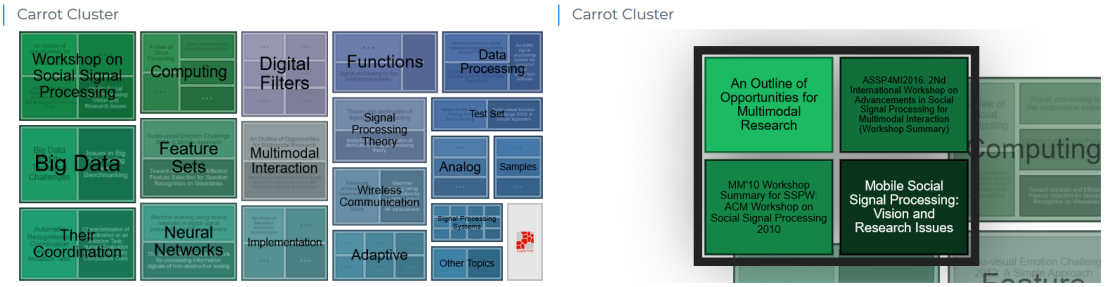


Abbildung 4.5: Visualisierung der Cluster nach einer Suche

Beim Laden der Seite rufen wir zunächst `foamtree = new CarrotSearchFoamTree()` welches die Visualisierung mit unseren Einstellungen initialisieren soll, diese Optionen werden mit einem „plain JavaScript object“[2] übergeben. Die Initialisierung generiert allerdings noch keine Visualisierung, dies geschieht erst in der Funktion `create_foam_tree`, welche wiederum die Methode `foamtree.set()` aufruft, die als Eingabe wieder ein „plain JavaScript Objects“ erhält bestehend aus den vom Carrot[2] plugin erstellten Clustern und den gefundenen Dokumenten innerhalb der Cluster. Nachdem die `set` Methode aufgerufen wurde wird auch direkt die Visualisierung mit unseren Daten geladen, wie sie in Abbildung 4.5 zu sehen ist.

4.2 Erstellen von Mappings für Elasticsearch

Mappings in Elasticsearch ist eine Funktion, welche erlaubt Elasticsearch vor dem Indizieren von Daten mitzuteilen, welche Form die Daten haben werden. Um ein Mapping zu erstellen, ist ein einfacher HTTP Aufruf auf den zu erstellendem Index REST Endpunkt nötig. In den Datensätzen existieren drei Felder welche abgebildet werden sollen, zwei Datumsfelder „issued“ und „accessed“ und das Textfeld „abstract“. Elasticsearch verwendet standardmäßig das `keyword` Datenformat, welches für Aggregation und einzigartige Strings wie Titel und E-Mailadressen oder Tags verwenden werden soll. Für einen Volltext wie zum Beispiel „abstract“ ist es allerdings ungeeignet.
Wir verwenden folgendes Mapping in unserer Anwendung:

```
1  {
2    "mappings": {
3      "entry": {
4        "properties": {
5          "issued": {
6            "type": "date",
7            "format": "yyyy-MM-dd||yyyy-MM||yyyy"
8          },
9          "accessed": {
10           "type": "date",
```

2. Ein simples JavaScript Objekt welches nur aus Schlüssel-Wert-Paaren besteht

```
11              "format": "yyyy-MM-dd||yyyy-MM||yyyy"
12            },
13            "abstract": {
14              "type": "text"
15            }
16          }
17        }
18    }
19 }
```

Dies ist das JSON Objekt, welches wir an den Server Endpunkt `/zotero/` des elasticsearch Servers senden. Wenn der elasticsearch Server also auf `localhost:9200` ist wäre die URL wie folgt `localhost:9200/zotero/`. Innerhalb dieses JSON Objektes setzen wir das Datumsformat auf drei mögliche „Datestring“ Formate, welche einfach nur drei mögliche Formate des ISO 8601[3] Standards sind.
Um dieses Mappings nun in Elasticsearch einzustellen, muss eine HTTP PUT Anfrage an den Endpunkt mit diesem JSON Objekt gesendet werden, dies ist mit jedem beliebigen HTTP Client möglich oder sogar in den „Dev Tools“ von Kibana. In unserem Beispiel werden wir nun das Kommandozeilen Programm `curl` verwenden.

```
curl -XPUT "localhost:9200/zotero"\
-H "Content-Type: application/json"\
--data '{ "mappings": { "entry":{ "properties": { "issued" : { "type": "
    date", "format": "yyyy-MM-dd||yyyy-MM||yyyy" }, "accessed" : { "type":
     "date", "format": "yyyy-MM-dd||yyyy-MM||yyyy" }, "abstract":{ "type":
     "text" } } } } }'
```

4.3 Erstellen eines Clusters via Carrot2 Plugin

Das Carrot2 plugin für elasticsearch registriert einen neuen REST Endpunkt in elasticsearch an der Ressource `localhost:9200/{index}/{type}/_search_with_clusters` welcher per GET oder POST Befehl aufgerufen werden kann. In diesem Projekt verwenden wir die GET Methode, in welcher wir folgende Parameter verwenden:

field_mapping_: verschiedene Zuordnungen zu Parametern die für den Clusteralgorithmus wichtig sind, in dem Projekt verwenden wir `field_mapping_content` und `field_mapping_title` zum Zuordnen des Textinhalts und des Titels (notwendig)

q: Kurz für query, dies ist der Suchbegriff nachdem die Dokumente durchsucht werden. (notwendig)

algorithm: Der Algorithmus, der verwendet werden soll. Standardmäßig wird der „Lingo“ Algorithmus verwendet. (optional)

size: Diesen Parameter legt fest wie groß das Suchfenster sein soll. Wir verwenden dies auf Empfehlung der Plugin Dokumentation in der beschrieben wird, dass das Suchfenster wenigstens 100 Dokumente groß sein soll. (optional)

3. Wolf und Wicksteed, *Date and Time Formats*.

Eine Beispielanfrage an das Carrot[2] Plugin mit dem Begriff „digital“ für die Suche in den Feldern `title` und `abstract` in den Dokumenten, kann durch folgenden GET Request abgerufen werden.

```
localhost:9200/zotero/entry/_search_with_clusters?q=digital&size=100&
    field_mapping_content=_source.abstract&field_mapping_title=_source.
    title&algorithm=lingo
```

Der Response der Abfrage entspricht einem JSON Objekt, dass die Metadaten, Cluster und Dokumente über den Suchbegriff enthält. Das Beispiel kann durch folgenden jQuery AJAX Aufruf implementiert werden:

```
1     var data = {
2         q : search_str,
3         field_mapping_title : "_source.title",
4         field_mapping_content : "_source.abstract",
5         size : "100",
6         algorithm: algorithm_str
7     };
8     return $.ajax({
9         method: "GET",
10        url : baseUrl + "/zotero/entry/_search_with_clusters",
11        data : data
12    });
```

5. Fazit und Ausblick

Das Ziel, ein System zum Clustering von Dokumenten zu entwickeln, konnte erreicht werden.

Im Rahmen des Projektes wurde eine Literatursammlung mit Metadaten von Dokumenten durch Zotero erstellt. Unser selbst entwickelte Java Anwendung hilft, die Bibliothek detaillierter und spezifischer zu analysieren und zu interpretieren. Durch die in Kibana erstellten Visualisierungen, die auch als exportierte JSON Datei zur Verfügung stehen, kann ein gesamtheitlicher Überblick der Dokumente erlangt werden. Die Implementierung eines Web-Servers ermöglicht eine Orchestrierung der eingesetzten Technologien. Er ermöglicht außerdem eine effiziente Such- und Analyse-Funktion und bildet die Ergebnisse in einem ansprechenchen Design ab.

Unsere neu gewonnenen Kenntnisse im Fachlichen und die Vorgehensweisen des Projektmanagements konnten wir in der Praxis vertiefen. In das äußerst interessante Thema Clustering von Dokumenten konnten wir wertvolle Einblicke gewinnen. Wir hatten außerdem die Möglichkeit mächtige Werkzeuge kennenzulernen, welche uns teilweise zuvor noch unbekannt waren. Besonders Elasticsearch als wertvolles Werkzeug bezüglich der Echtzeitverarbeitung von großen Datenmengen, stellte sich für uns als sehr nützlich heraus. In unserer akademischen Laufbahn wird es zukünftig an anderer Stelle Anwendung finden.

Durch die Einhaltung von Best-Practices während der Projektumsetzung und dem Aufsetzen des Java-Servers, ist eine effiziente Wartung und eine unproblematische Weiterentwicklung möglich.
Ein zukünftiges Funktion wäre beispielsweise ein direkter Dateiupload der exportierten Zotero JSON-Datei. Durch die Integration unseres Java-basierten Konvertierungstools in den Web-GUI-Server ist eine direkte Verarbeitung der Daten bis hin zur Darstellung von Clustern möglich.
Wenn weitere interessante Zusammenhänge in den Daten offenlegt werden, kann die Kibana Visualisierungsseite einfach durch das Hinzufügen von neuen `iframe` Elementen ergänzt werden (Share Button auf der Kibana Plattform).

Mit Freude haben wir an diesem Semesterprojekt gearbeitet und sind sehr zufrieden mit dem Endergebnis. Wir freuen uns darauf, das Ergebnis dieses Projektes in der finalen Projektpräsentation vorzustellen und zusammenzufassen.

Literaturverzeichnis

Bulk API — Elasticsearch Reference [6.1] — Elastic. Learn/Docs/Elasticsearch/Reference/6.1. Besucht am 28. Januar 2018. https://www.elastic.co/guide/en/elasticsearch/reference/current/docs-bulk.html.

Foundation, jQuery. *jQuery*. Besucht am 25. Januar 2018. http://jquery.com/.

Kibana 5.5.2. Besucht am 26. Januar 2018. https://www.elastic.co/downloads/past-releases/kibana-5-5-2.

Developer-first Project Management for Teams on GitHub. Besucht am 25. Januar 2018. https://waffle.io.

Wolf, Misha, und Charles Wicksteed. *Date and Time Formats*. Besucht am 27. Januar 2018. https://www.w3.org/TR/NOTE-datetime.

A. Abkürzungsverzeichnis

DBPRO = Datenbankprojekt
IMPRO = Informationsmanagementprojekt
HTTP = Hypertext Transfer Protocol
HTML = Hypertext Markup Language
REST = Representational State Transfer
API = Application-Programming-Interface
ES = Elasticsearch
CSL JSON = Citation Style Language JSON schema
JSON = JavaScript Object Notation
JS = JavaScript
CORS = Cross-Origin Resource Sharing
DOM = Document Object Model
AJAX = Asynchronous JavaScript And XML.

Dokumenten Clustering DBPRO

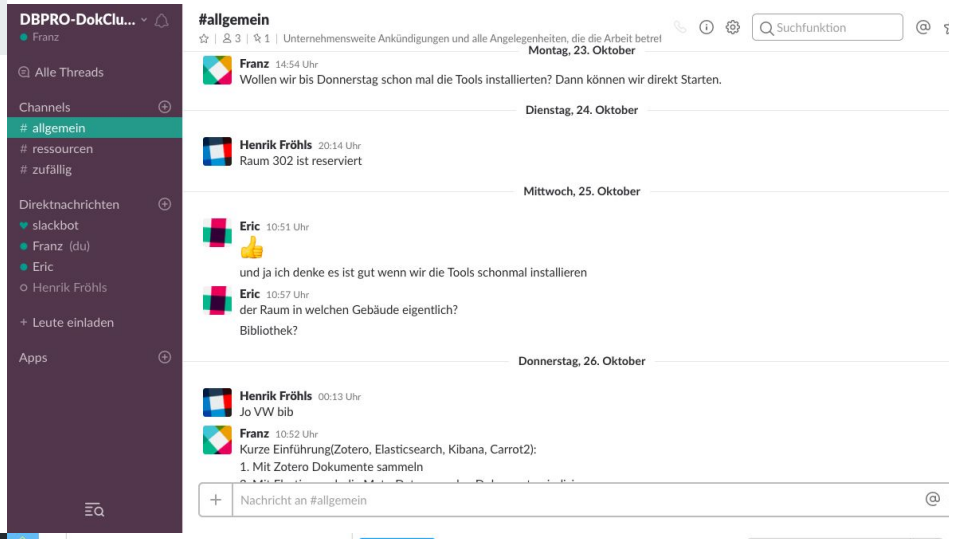
Betreuer: Holmer Hemsen, Technische Universität Berlin, 13.11.2017

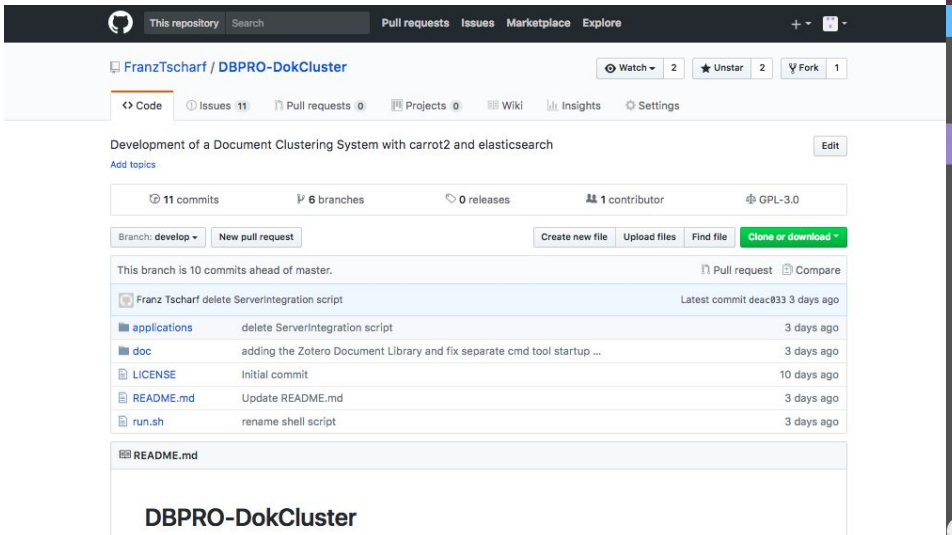
Gliederung

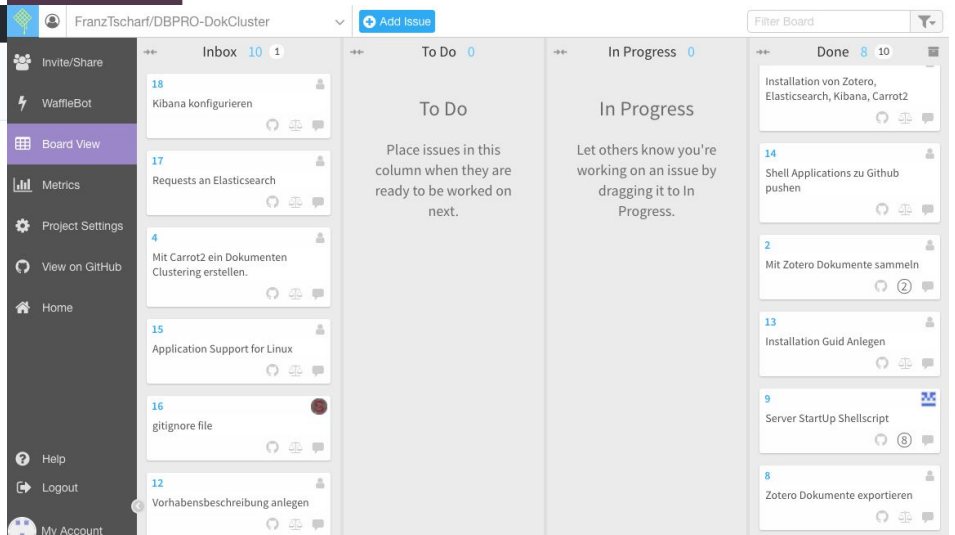
1. Management
2. Zielsetzung
3. Umsetzung
4. Dokumentation

1. Projektmanagement

- Kommunikation über Slack
- Repository auf GitHub „DBPRO-DokCluster“
- Arbeits- und Issue-Koordination mittels Waffle







2. Projektzielsetzung

- Entwicklung eines Systems
- Für Indexierung, Clustering und Suche von Dokumenten
- Visualisierung durch eine Web GUI

3. Projektumsetzung -Tools

- zotero
- elastic
- carrot2
- kibana

3.1. zotero

zotero

- Bibliothek für Artikel, Bücher, Webseiten, Manuskripte, Dokumente… erstellen
- Metadaten werden automatisch importiert
- Quellen werden beim Browsen direkt erkannt und können gespeichert werden

3.2. elasticsearch

elastic

- Search Server
- NoSQL Datenbank
- Interaktion durch RESTful API
- JSON

3.2. JSON

```
{ "name" : "LGNx4mW",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "PTxhvyB9ReCctS_jCHiOTA",
  "version" : {
    "number" : "5.5.2",
    "build_hash" : "b2f0c09",
    "build_date" : "2017-08-14T12:33:14.154Z",
    "build_snapshot" : false,
    "lucene_version" : "6.6.0"
  },
  "tagline" : "You Know, for Search"
}
```

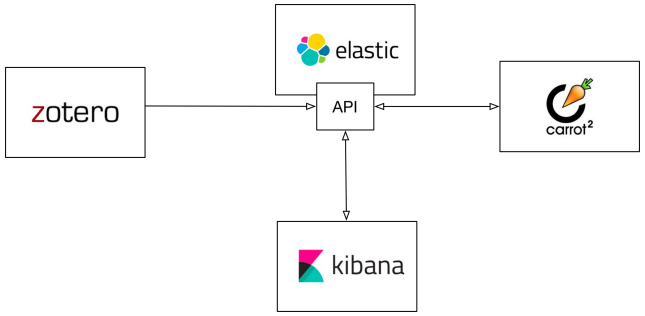
3.3. Carrot²

- Open Source Software
- Kollektionen von Dokumenten (z.B. Suchergebnisse) in Cluster aufteilen

3.4. kibana

- Plugin zur Visualisierung von Elasticsearch Ergebnissen

3.5. Architektur



4. Projektdokumentation

- Meilenstein Präsentationen.
- Dokumentation über GitHub Wiki.
- Abschlussreport in LaTeX über Overleaf.

Danke!
...Fragen?

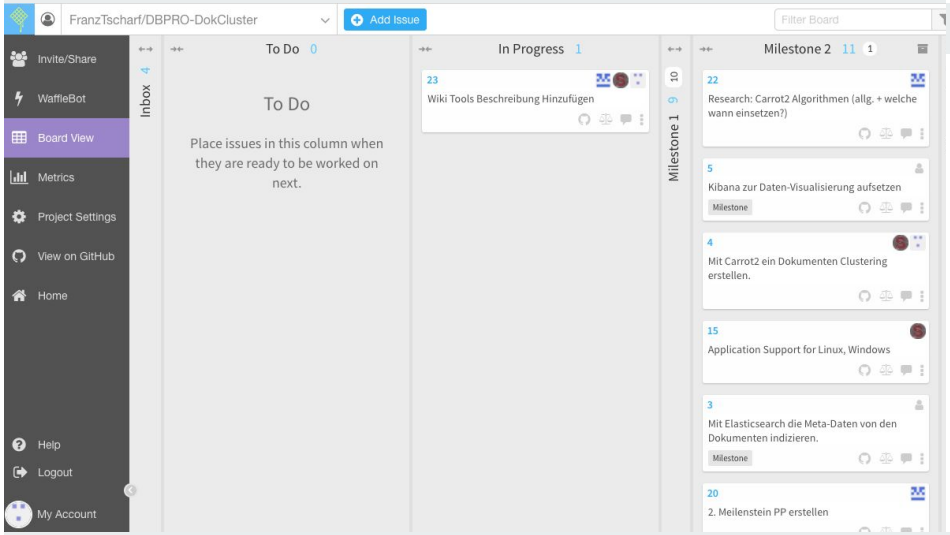
Team: Henrik Fröhls, Erik Schneider, Franz Tscharf

Dokumenten Clustering DBPRO

Meilensteinpräsentation 2.
Betreuer: Holmer Hemsen, Technische Universität Berlin, 11.12.2017

Gliederung

1. Fortschritt (waffle.io)
2. Strukturierung
3. Umsetzung
4. Live Demo
5. Ausblick



Strukturierung

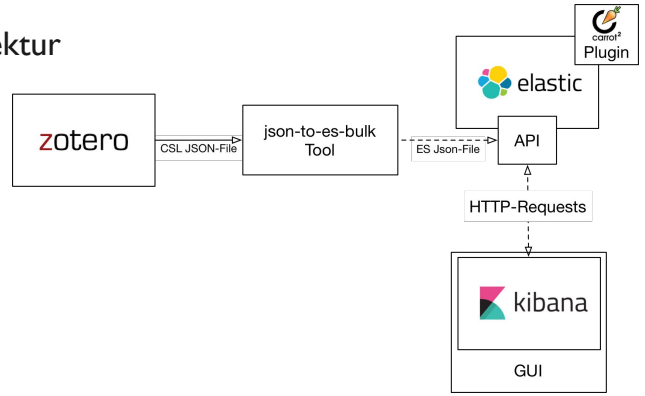
- Alle notwendigen Tools & Server strukturiert, konfiguriert & organisiert
- Travis CI Pull Request Überprüfung

Startup Shell Script

run.sh

- Zentrales Script, mit dem alle Server deployed werden können
- Plattformunabhängig

Architektur



Carrot² Algorithmen

- Lingo Algorithmus
- Suffix Tree Clustering (STC) Algorithmus
- K-Means Algorithmus

Lingo

1. Cluster Labels identifizieren
   a. Term-Document-Matrix erstellen
2. Dokumente Labels zuweisen

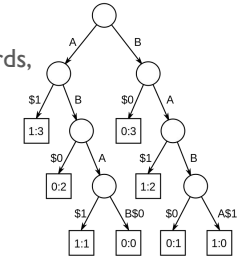
Lingo: 1.a.: Term-Document-Matrix

- Beispiel: 2 kurze Dokumente:
  - D1 = “I like Database Systems.”
  - D2 = “I hate Database Systems.”
- Term-Document Matrix:

	I	like	hate	Database	Systems
D1	1	1	0	1	1
D2	1	0	1	1	1

STC - Suffix Tree Clustering

1. Generalized Suffix Tree (GST) erstellen
2. GST durchsuchen nach sich wiederholenden key-words, um Base Cluster zu erstellen
3. Base Cluster werden zu Clustern kombiniert



k-Means

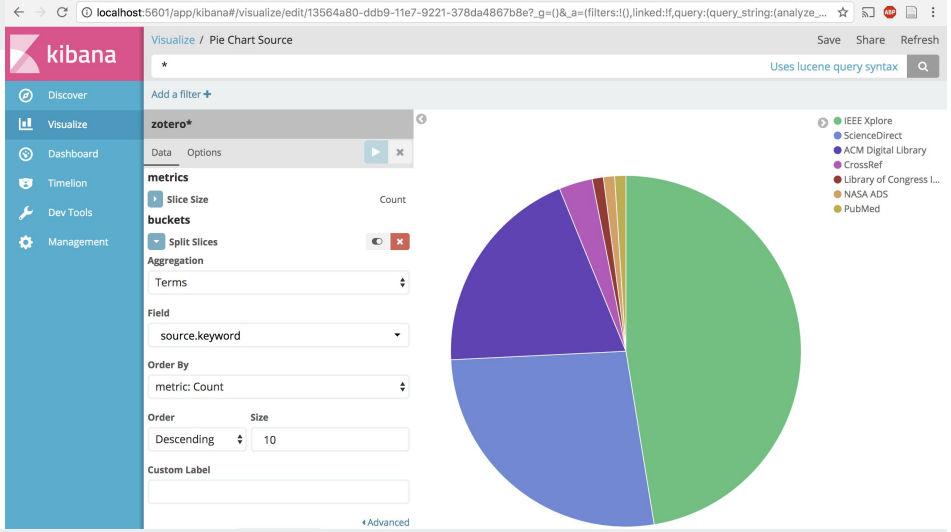
- Funktionsweise ähnlich wie Lingo
- Produziert nicht überlappende Cluster

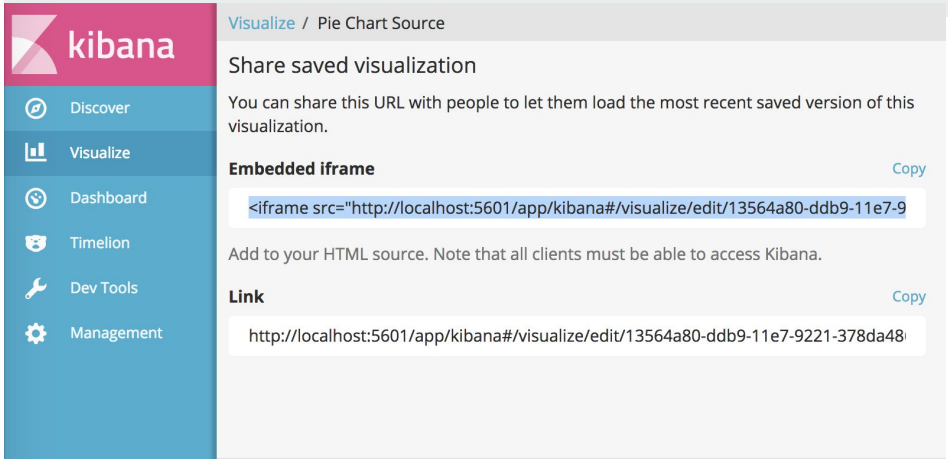
Wann welchen Algorithmus verwenden?

	Lingo	STC	k-means
Wohl-geformte längere Labels benötigt	x		
Viele kleine Cluster benötigt	x		
Großes Dokumenten Set (Hohe Performance erforderlich)		x	
Nicht überlappende Cluster benötigt			x

Web-GUI Entwicklung

- Framework: Bootstrap
- JS Library FoamTree für Cluster Suche
- Erstellen von Visualisierungen durch Kibana





Das Carrot² elasticsearch Plugin

- Über `/{index}/{type}` angesprochen
- Zusätzlich `_search_with_clusters` am ende
- Beispiel
  - `http://localhost:9200/zotero/entry/_search_with_clusters?q=digital`
- `GET` und `POST` möglich, `POST` empfohlen von Plugin

Das Carrot² elasticsearch Plugin

- Braucht außerdem ein fieldmapping
  - Titel und Inhalt müssen zugeordnet werden für clustering Algorithmus
- Gibt JSON zurück mit indizierten Dokumenten und Clustern

Live-Demonstration

Ausblick

- Erweiterung & Feintuning
- aws - Hosting
  + Schnell, sicher, zuverlässig
  + Flexible Skalierung
  + Pay-Per-Use
  + Amazon ES Service

Danke!
...Fragen?

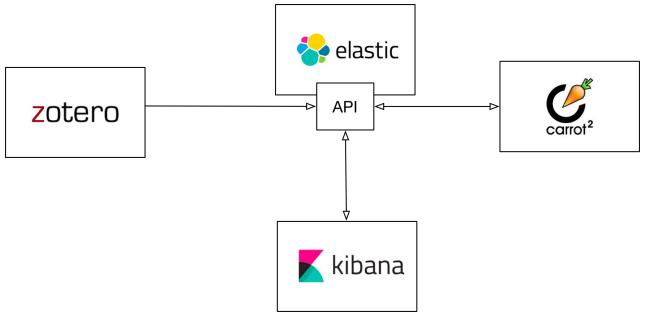
Team: Eric Schneider, Henrik Fröhls, Franz Tscharf

Visualisierung durch Carrot2

-Mithilfe von Javascript kann eine Clustersuche Visualisiert werden.

-Code snipped von FoamTree zur integration vom Cluster

3.5. Architektur



Zotero-Dokumente to Elasticsearch

- Json-to-es-bulk
  - Konvertiert Zotero-Output zu Elasticsearch Input

Visualisierung durch Kibana

-was ermöglicht kibana?

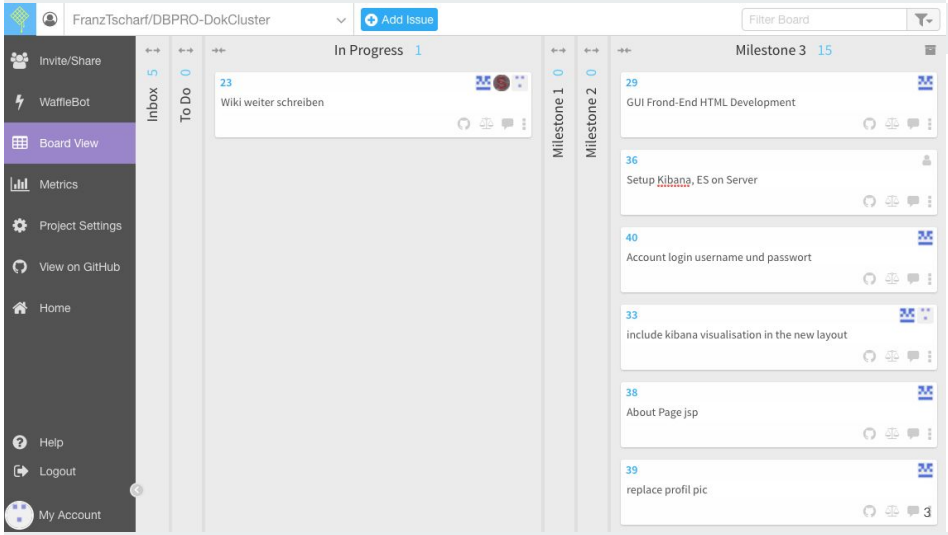
-wie fügen wir das Diagramm hinzu?

Dokumenten Clustering DBPRO

Meilensteinpräsentation 3.
Betreuer: Holmer Hemsen, Technische Universität Berlin, 22.01.2018

Gliederung

1. Fortschritt (waffle.io)
2. Herausforderungen
3. Umsetzung
4. Carrot-Konfiguration
5. Konvertierungstool
6. Live Demo



2. Herausforderungen

- Web-Server für die Grafische Darstellung und Persistenz von Daten.
- Json Konvertierungstool durch Java

3. Umsetzung Web-Server

- GUI Web-Server
  - Java Spring Web MVC
  - Dependency Management mit Maven
  - Server über Tomcat 8.5
  - Authentifikation über Spring Security

4.1. Stopwords

- Unaussagekräftige Wörter filtern
- "../config/elasticsearch-carrot2/stopwords.en"
- "Chapter", "Tags", "I", "We"
  - Diese Wörter werden beim Clustering ignoriert

4.2. Auswahl des Clusteralgorithmus

info: {...}
algorithm: lingo
search-millis: 3
clustering-millis: 51
total-millis: 55

info: {...}
algorithm: stc
search-millis: 4
clustering-millis: 19
total-millis: 24

info: {...}
algorithm: kmeans
search-millis: 6
clustering-millis: 15
total-millis: 22

5. Umsetzung Konvertierungstool

- ehem. json-to-es-bulk
- Java Implementierung (erleichtert Spring-Einbindung)
- "Übersetzt in ES-fähiges JSON-Format
  - Autoren als Arrays
  - Datum in ES-fähiges Format


```
{
[...]
"author": [
  {
    "family": "Djigan",
    "given": "V. I."
  },
  [...]
],
"issued": {
  "date-parts": [
    [
      "2017",
      9
    ]
  ]
}
```



```
{"index":{"_index":"zotero","_type":"entry","_id":1}}
{[...],"author":["Djigan, V. I.","Shakhtarin, B.
I.","Likhoedenko, K. P.","Zhurakovskiy, V. N."],
[...] "issued":"2017-09", [...]}
```


Live-Demonstration

Danke!
...Fragen?

Team: Eric Schneider, Henrik Fröhls, Franz Tscharf

Ausblick

- Erweiterung des Webservers auf eine ES Konfigurations Page
- JSON CSL File Upload
- Automatisiertes Indexieren
- ES und Kibana URL als Konfigurationsoption