

# elasticsearch



Pawel Kowalski & Marco Schmalz  
iterativ.ch

Sept 2016

# **Suche & Information Retrieval**







Zafavia, River.  
Bassett, Dr. store at Mackay.  
Port Mackay, Roth.

Collinson Index  
BAM - BRIS

# Use Cases für Elasticsearch

## Custom search engine (-toolbox)

- Globale Suche in Webapps

## Event-store für Sysadmins, Security Monitoring

- Zusammen mit Kibana - vergleichbar mit Splunk

## NoSQL Datenbank

- Eine einfach NoSQL DB für json Dokumente

# Das Problem



Google Search

I'm Feeling Lucky

Google.ch offered in: [Deutsch](#) [Français](#) [Italiano](#) [Rumantsch](#)

# Themenübersicht

- Wie wird die Indexierung und Suche implementiert
  - Normalisierung & Token-vergleiche
  - Abfragen
  - Relevanz
- Datenstrukturen für die Suche
- Aggregationen
- Universelle Suche aus der UX Perspektive

# Theorie & Themen

## Information Retrieval

- TF/IDF
  - Term Frequency
  - Inverse Document Frequency
- Precision / Recall
- Normalisierung für den Vergleich
  - Tokenizing
  - Lowercase
  - Stemming
  - Soundex
  - Stopwords (to be or not to be)
  - Decomposition (für Deutsch)
  - Synonyme

## Implementation

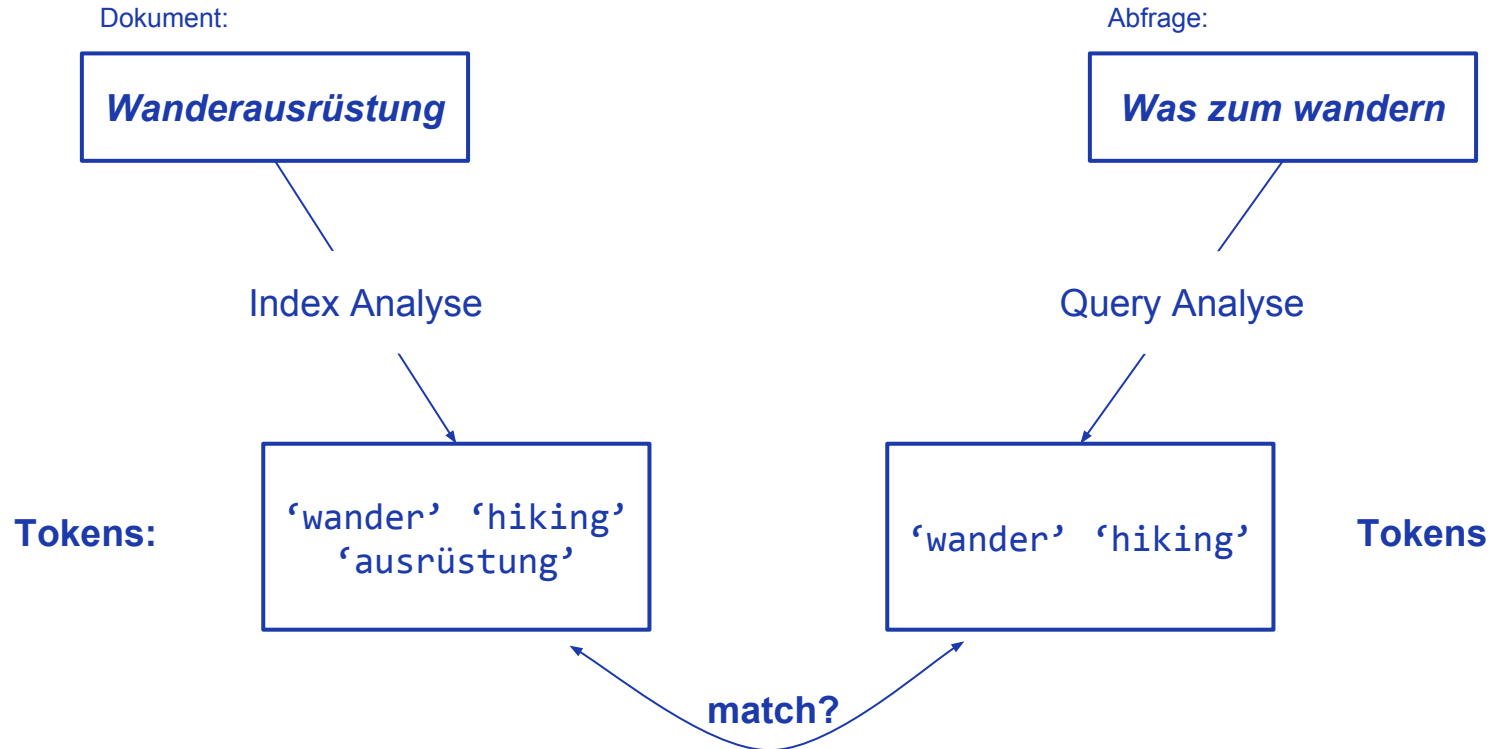
- Querying
- Inverted Index
  - Was ist das? Ihr kennt das alle: jmd aus dem Publikum auspicken
- Aggregations

## UX

- Universelle Suche: über alle Felder



# Normalisierung - Vergleichbarkeit



# Normalisierung - Vergleichbarkeit

Damit die Texte verglichen und gefunden werden können:

Es braucht eine Grundform. Ansonsten würde der Index mit den verschiedenen Formen riesig, oder wir hätten sehr viele Vergleiche

- Tokenizing - Text in tokens aufsplitten
- Stemming - Stammformen der Wörter
- Stopwords - wie, auch, und, etc.. tragen *nicht* zur Information bei..
- Decomposition - Zusammengesetzte Wörter auf Deutsch
- Synonyme - Bürosessel vs. Bürostuhl

# Normalisierung - Vergleichbarkeit

**Query:** *Suche spannende Routen zum Wandern oder **Velofahren***

**tokens:** ['Suche', 'spannende', 'Routen', 'zum', 'Wandern', 'oder', '**Velofahren**']

**lower:** ['suche', 'spannende', 'routen', 'zum', 'wandern', 'oder', '**velofahren**']

**stop:** ['suche', 'spannende', 'routen', 'wandern', '**velofahren**']

**stem:** ['such', 'spannend', 'rout', 'wander', '**velofahr**']

**decomp:** ['such', 'spannend', 'rout', 'wander', '**velofahr**', '**velo**', '**fahr**']

**synonyme:** ['such', 'spannend', 'rout', 'wander', '**velofahr**', '**velo**', '**rad**', '**fahr**']

# Normalisierung - Vergleichbarkeit

**Query:** *Suche spannende Routen **zum** Wandern oder Velofahren*

**tokens:** ['Suche', 'spannende', 'Routen', '**zum**', 'Wandern', 'oder', 'Velofahren']

**lower:** ['suche', 'spannende', 'routen', '**zum**', 'wandern', 'oder', 'velofahren']

**stop:** ['suche', 'spannende', 'routen', 'wandern', 'velofahren']

**stem:** ['such', 'spannend', 'rout', 'wander', 'velofahr']

**decomp:** ['such', 'spannend', 'rout', 'wander', 'velofahr', 'velo', 'fahr']

**synonyme:** ['such', 'spannend', 'rout', 'wander', 'velofahr', 'velo', 'bike', 'fahr']

# Decomposition

## Guter Beispiel zu dem wann die Normalisierung passiert

Zur Index-Zeit:            Beim indexieren wird der text normalisiert

Zur Abfrage-Zeit:        Beim abfragen wird die query normalisiert

**Klobürste - Zahnbürste**     $\Rightarrow$  ['klo', 'bürste'], ['zahn', 'bürste']

## Deshalb:

Index-Zeit:    Decomposition             $\Rightarrow$  [0:'zahn', 1:'bürste', 2:'zahnbürste']

Abfrage:       Keine Decomposition     $\Rightarrow$

'Zahnbürste'  $\Rightarrow$  match mit    2:'*zahnbürste*'

'Zahn bürste'  $\Rightarrow$  match mit    0:'*zahn*' und 1:'*bürste*'

'Bürste'             $\Rightarrow$  match mit    1:'*bürste*' d.h. Klob*ürste* und Zahn*bürste*

# Decomposition

Verschiedene Ansätze wie die Decomposition implementiert werden kann:

- Dictionary basiert
- The elasticsearch decomposer uses prebuilt Compact Patricia Tries

Teleobjektiv           ⇒       [‘tele’, ‘objektiv’]

Telezoomobjektiv    ⇒       [‘tele’, ‘zoom’, ‘objektiv’]

Urinstinkt            ⇒       [‘ur’, ‘instinkt’]   oder   [‘urin’, ‘stinkt’]



# Synonyme

- Expansion

Velo	Fahrrad	Bike	MTB
Lautsprecher	Boxen	Speaker	

- Canonical Form

Notebook	Laptop	Portable PC	⇒	Laptop
Velo	Fahrrad	Bike		Velo

Bei der 'Canonical Form' können Kollisionen mit anderen Begriffen vermieden werden wie zB. 'Portable PC' ⇒ Natel aka. 'Portable'

# Abfragen

Was machen wir mit der Eingabe des Benutzers?

- Suchmodus: Full-text (Google) vs. Kriterien-Suche (Shop)
- Alle Begriffe (AND) oder einzelne (OR)
- Etwas dazwischen:  $n-i$  aus  $n$  ( $i < n$ )

**Liefert die Suche möglichst immer ein Resultat?**

vs.

**Dürfen nur Resultate die den Suchkriterien entsprechen angezeigt werden?**

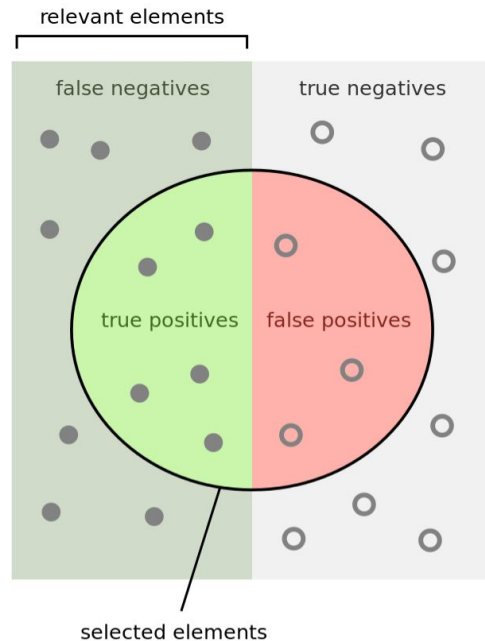
# Precision & Recall

Precision:

$$\text{Precision} = \frac{tp}{tp + fp}$$

Recall:

$$\text{Recall} = \frac{tp}{tp + fn}$$

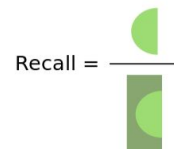


How many selected items are relevant?



$$\text{Precision} = \frac{\text{green}}{\text{green} + \text{red}}$$

How many relevant items are selected?



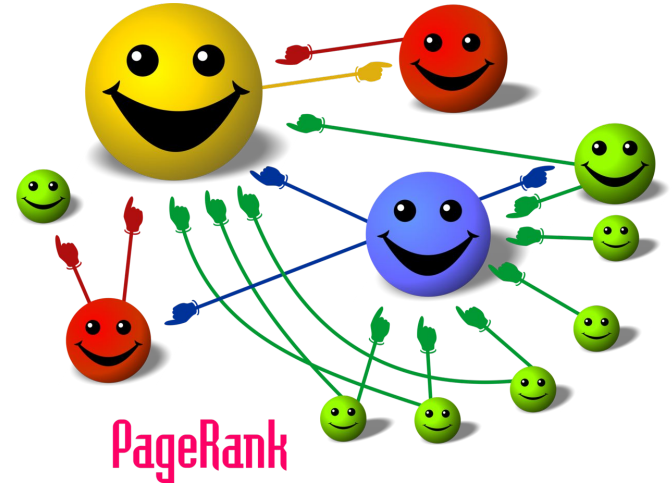
$$\text{Recall} = \frac{\text{green}}{\text{green} + \text{dark gray}}$$

# Ranking

## Was kommt zuoberst?

Bekannte Algorithmen:

- Googles Page-Rank (Hypertext)
- Reddit Ranking (Fokus auf Votes)
- HackerNews Ranking (Fokus auf Neuheit)
- Machine learned ranking (Click-Verhalten der Benutzer als Input)
- TF/IDF (Probabilistisch)



# Ranking - TF / IDF

- Term Frequency / Inverse Document Frequency

Einfache Heuristik zur Berechnung der Relevanz eines Dokuments bezüglich der eingegebenen Suchgriffe.

## Term Frequency:

$$TF(t) = \frac{\text{(Häufigkeit des Begriffs } t \text{ im Dokument)}}{\text{(Anzahl Wörter im Dokument)}}$$

## Inverse Document Frequency:

$$IDF(t) = \log_e \left( \frac{\text{Anzahl der Dokumente}}{\text{Anzahl der Dokumente mit dem Begriff } t} \right)$$

# Inverted Index

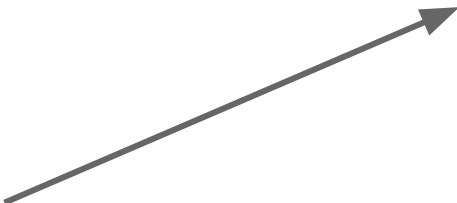
ID	Text
----	------

- 1 Schwimmen wir auch im Winter in der Aare?
- 2 Im Winter sind wir in Silvaplana
- 3 Aare ist wunderbar



Dokumente:

Inverted Index:



Term	Freq	Document IDs
schwimmen	1	[1]
wir	1	[1] [2]
auch	1	[1]
im	2	[1] [2]
winter	2	[1] [2]
in	2	[1] [2]
der	1	[1]
aare	2	[1] [3]
sind	1	[2]
silvaplana	1	[2]
wunderbar	1	[3]



# Aggregationen

- Sind sehr schnell (siehe Inverted Index)
- Können auch geschachtelt sein
  - zB Altersstruktur pro Stadt
- Früher wurde das 'Facettensuche' genannt
  - zB. Bei Suche der Leute:
    - Stadt
    - Alter

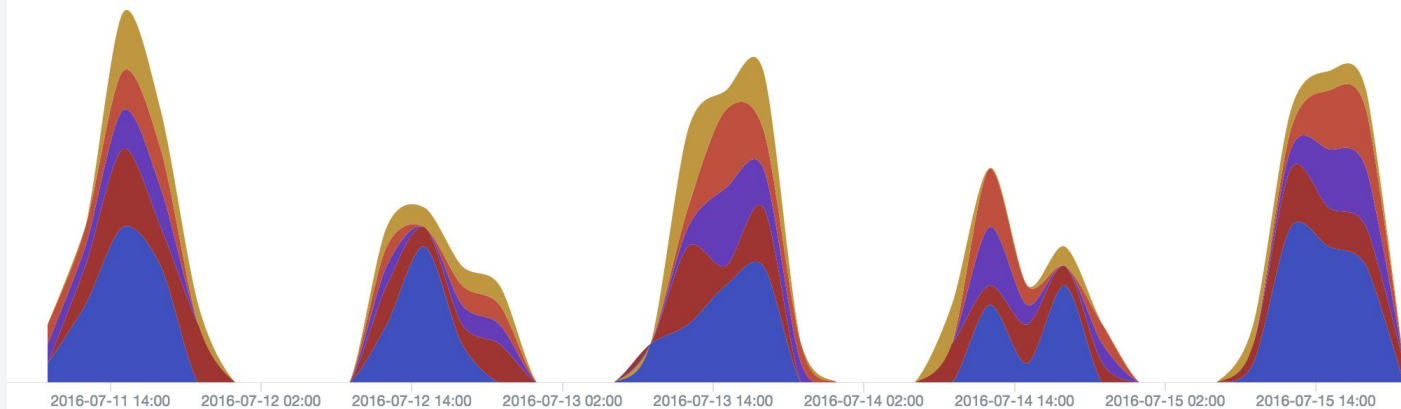
## Typen:

- Terms, Histogram, Geo, Dates, Significant Terms
  - <https://www.elastic.co/blog/significant-terms-aggregation>

# Aggregationen

## KATEGORIEN

- Aftershave & Rasurpflege (12)
- Body Lotion & Body Butter (5)
- Business Schuhe (26)
- Champagner & Schaumwein (1)
- Deo & Antitranspirant (10)
- Duschgel & Duschschaum (8)
- Körperpflegeset (15)
- Parfum (73)
- Rotwein (6)
- Schul- & Lernbücher (2)
- Süsswein (1)
- Weisswein (1)



## Ergebnisse für «lego»

Preis



CHF 0.00 - 20.00  
CHF 20.00 - 50.00  
CHF 50.00 - 100.00  
CHF 100.00 - 200.00

Marke



Händler



# UX - Universelle Suche

MyService CRM

Suchen

Neukunde erfassen

Kunde importieren

## Suchen

🔍 Suchen

Nach Name, BSK-Nummer, Telefon, Firma und Ort


**siroop.ch**

Wonach suchst du?



# UX - Universelle Suche - nooot!

## Kundensuche

 Prepaid Kunden können nur mit Telefon- oder SIM-Nr. gesucht werden!

Bitte geben Sie Suchkriterien

Telefonnummer

SIM-Nr.

SCN

E-Mail

Vorname

Name /  
Firmenname

Firmenindex-Nr.

Geburtsdatum  .  .

Strasse Nr.

☐ Keine Strassenbezeichnung

PLZ / Ort

Land  

Suchen

# Praktischer Einsatz


Server

localhost:9200

```
1 POST /multiwords/words/1
2 {
3   "text": "tele objektiv in da house"
4 }
5
6 POST /multiwords/_search
7 {
8   "query": {
9     "dis_max": {
10      "tie_breaker": 0.7,
11      "boost": 1.2,
12      "queries": []
13    }
14   }
15 }
16 }
17
```

```
1 {
2   "count": 61958,
3   "_shards": {
4     "total": 5,
5     "successful": 5,
6     "failed": 0
7   }
8 }
```

# Praktischer Einsatz



## Sense (Beta)

offered by cheics


★★★★★ (133) [Developer Tools](#) 70,336 users

OVERVIEW REVIEWS SUPPORT RELATED

Server: 192.168.231.50:9200/telemetry-dev

```
6 GET _search
7 {
8   // All events actually have a doctype
9   query:{
10     "term":{"doctype":"event"}
11   }
12 }
13 GET _search
14 {
15   "explain": true,
16   "facets": {
17     "natlity": {
18       "partial_fields": {
19         "field": "k2"
20       }
21     }
22   }
23   search_type: "stats"
24   // Owen was using this to determine number
25   // of unique users... K2 cannot be used
26   // this way. It often has a value of
27   // things like "landscape"
28   // We get 49 users here
29 }
30 GET _search
31 }
```

```
1 {
2   "took": 2,
3   "timed_out": false,
4   "_shards": {
5     "total": 5,
6     "successful": 5,
7     "failed": 0
8   },
9   "hits": {
10    "total": 128560,
11    "max_score": 0.99996144,
12    "hits": [
13      {
14        "_index": "telemetry-dev",
15        "_type": "event",
16        "_id": "924148433984d17fbb3831acc251f1d4",
17        "_score": 0.99996144,
18        "_source": {
19          "_rev": "1-2372ccb8e867563e4386da7aac230c43",
20          "appId": "a1fwUmay0V00eZwR0ak49MbyU4u6e10",
21          "doctype": "event",
22          "_id": "924148433984d17fbb3831acc251f1d4",
23          "event": {
24            "ts": 1407402918,
25            "Type": "ValidateAccessTokenTelemetryEvent",
26            "IpAddress": "157.56.164.154",
27            "UserIdToken": "3bc32978bebe4456ab37def"
```


 Compatible with your device

### A JSON aware developer console to Elasticsearch.

Unofficial!

Forked from  
<https://github.com/bleskes/sense>, which is no longer present in the Chrome Extension Store

Now maintained (or not really maintained) here:  
<https://github.com/cheics/sense>

 [Report Abuse](#)

### Additional Information

Version: 0.9.0



# Praktischer Einsatz

## Dokument-Suche

- Indexierung
- Querying
- Aggregationen
- Suggestions

## Betriebliche & Setup-Aspekte

- Mappings
- Zero Downtime mapping-Änderungen: <http://elastic.iterativ.ch>
- Backups vs. MasterDB

# Praktischer Einsatz - Indexierung

POST /people/person/1

```
{  
  "name": "Pawel Kowalski",  
  "ort": "Bern",  
  "plz": "3014",  
  "strasse": "Tellstrasse 8",  
  "email": "paweloque@gmail.com"  
}
```

POST /people/person/2

```
{  
  "name": "Marco Schmalz",  
  "ort": "Bern",  
  "plz": "3012",  
  "strasse": "Melchtalstrasse 23",  
  "email": "marco@schess.ch"  
}
```

# Praktischer Einsatz - Querying

```
POST /people/_search
```

```
{  
  "query": {  
    "match": {  
      "name": "pawel"  
    }  
  }  
}
```

```
POST /people/_search
```

```
{  
  "query": {  
    "match": {  
      "ort": "bern"  
    }  
  }  
}
```

# Praktischer Einsatz - universelle Suche

```
POST /people/_search
{
  "query": {
    "bool": {
      "should": [
        {
          "match": {
            "ort": "Bern"
          }
        }, {
          "match": {
            "name": "Bern"
          }
        }
      ]
    }
  }
}
```

# Praktischer einsatz - Aggregationen

POST /people/\_search

```
{
  "size": 0,
  "aggs": {
    "strassen": {
      "terms": {
        "field": "strasse"
      }
    }
  }
}
```

POST /people/\_search

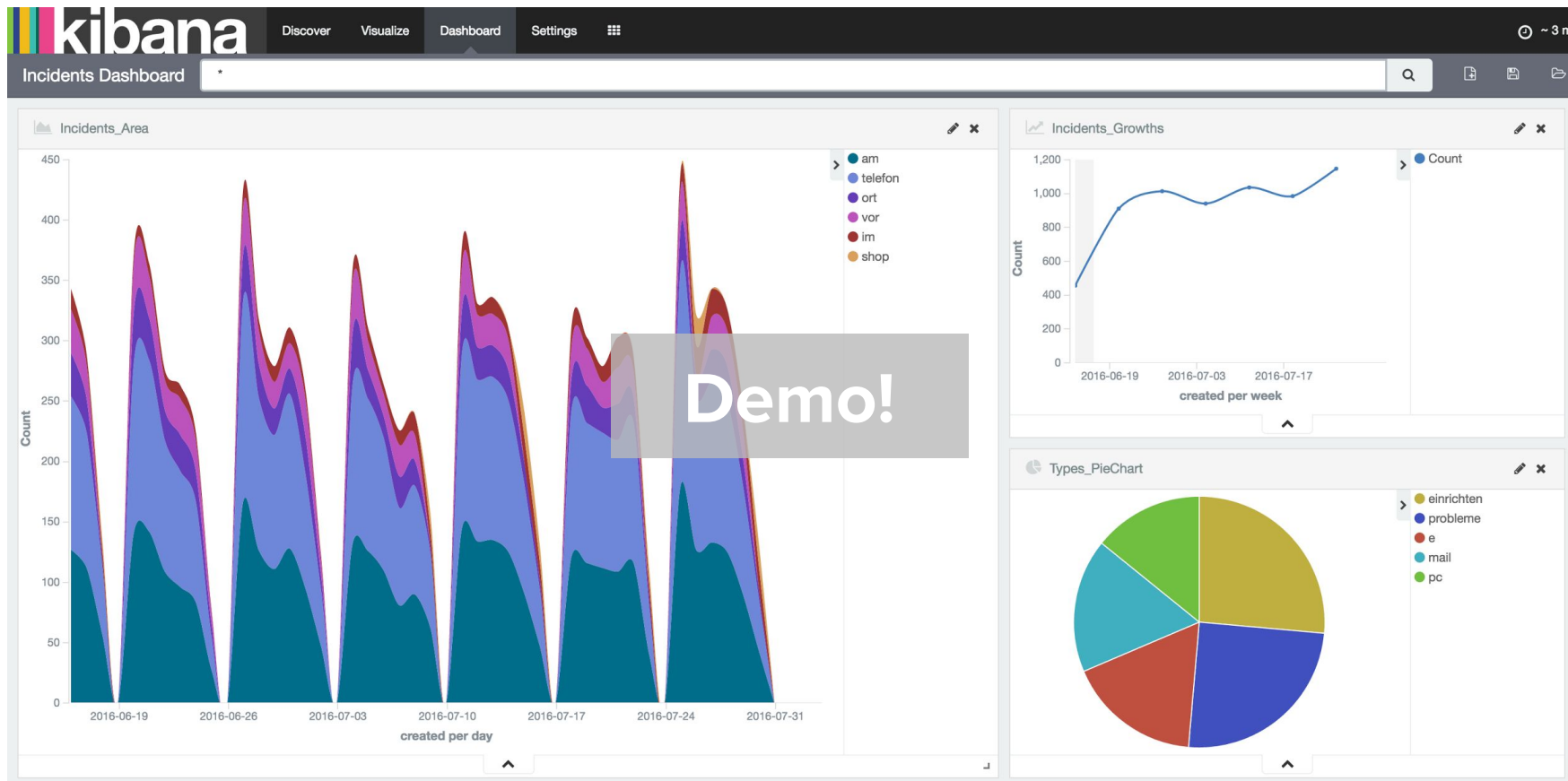
```
{
  "size": 0,
  "aggs": {
    "places": {
      "terms": {
        "field": "ort"
      }
    }
  }
}
```

# Praktischer Einsatz - Mapping

```
"decomp_analyzer": {
  "tokenizer": "decomp",
  "filter": [
    "lowercase",
    "keywords",
    "decomp",
    "synonym_filter",
    "german_stop",
    "german_normalization",
    "german_stemmer",
    "filter_secret_stop",
    "unique"
  ]
},
"name": {
  "type": "string",
  "analyzer": "main_analyzer",
  "copy_to": "full_text",
  "fields": {
    "decomp": {
      "type": "string",
      "analyzer": "decomp_analyzer",
      "Search_analyzer": "main_analyzer"
    }
  }
}
```



# Praktischer Einsatz - Kibana



# Spannende Themen - Beyond elastic



# Spannende Themen - Beyond elastic

**Feedback Loop** aus dem Benutzerverhalten - **alle Interaktionen loggen!**

(auch wenn es nicht klar ist, wie die Daten ausgewertet werden):

```
{
  "date_time": "2016-01-31 14:58:40",
  "session_number": 1,
  "source": {
    "q": "iphone",
    "category": "Handy",
    "page": "1"
  },
  "sink": {
    "q": "iphone 6s",
    "page": "1"
  },
  "device_id": "56addb26b49e12.74785018"
}
```

```
{
  "date_time": "2016-01-31 08:22:30",
  "session_number": 0,
  "source": {
    "q": "ipad mini",
    "category": "Tablet",
    "brand_name": "Apple",
  },
  "sink": {
    "q": "ipad mini",
    "category": "Tablet",
    "brand_name": "Apple",
    "sort": "price",
  },
  "device_id": "56adc4417ee674.11822684"
}
```

# Spannende Themen - Beyond elastic

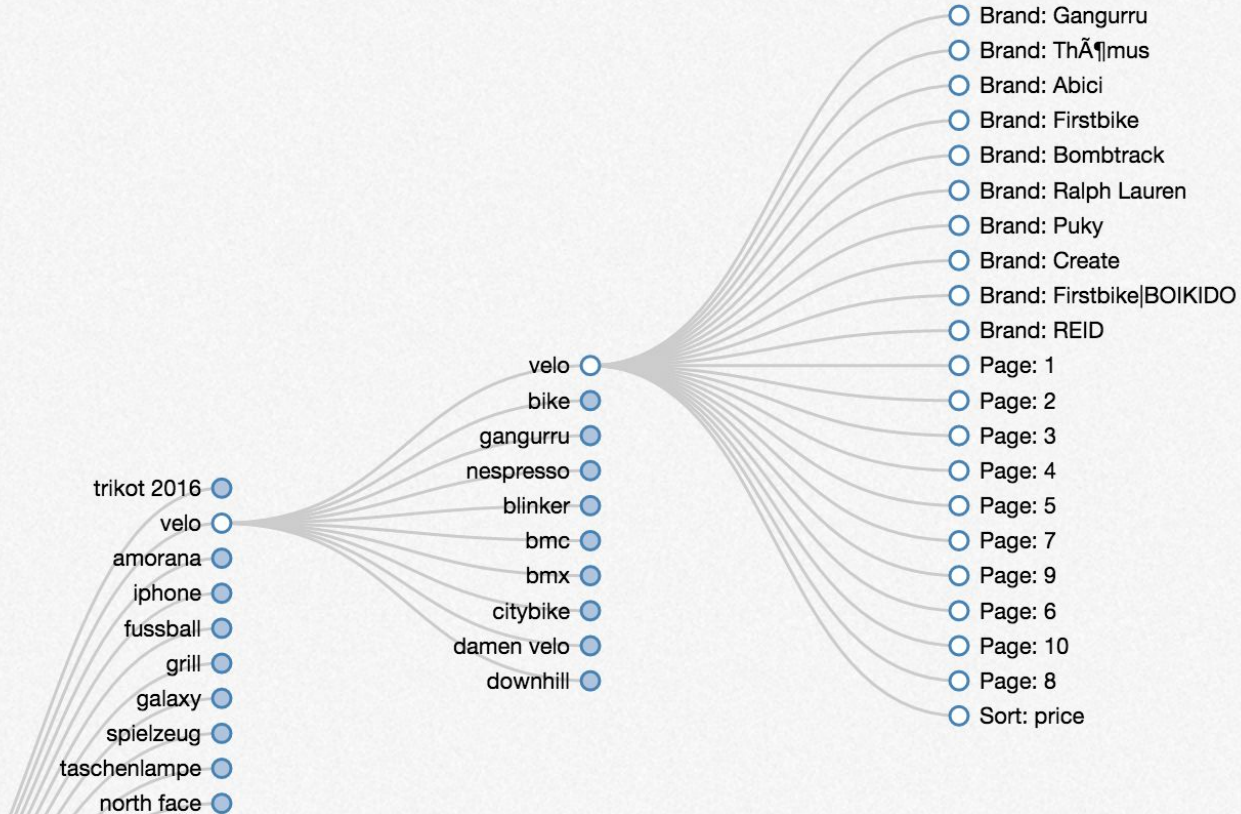
## Was machen wir mit diesen Daten?

- Term - Pfade
  - Lego - Lego Technic - Arocs - [produkt]
- Synonyme
  - Bürosessel - Bürostuhl
- Wahrscheinlichkeit für den nächsten Begriff
  - Grillzange - Grillgabel - Grillbesteck

# Spannende Themen - Beyond elastic



# Spannende Themen - Beyond elastic



# Spannende Themen - Beyond elastic

Input Preprocessing:

*Ich suche Mammut Hiking Ausrüstung für den Herbst*

Named Entity Annotation:

<b>Brand</b>	<b>Category</b>	<b>Jahreszeit</b>
↓	↓	↓
<i>Ich suche Mammut Hiking Ausrüstung für den Herbst</i>		

[https://en.wikipedia.org/wiki/Named-entity\\_recognition](https://en.wikipedia.org/wiki/Named-entity_recognition)

# Fazit

- Elasticsearch kann sehr vielseitig eingesetzt werden
- Es ist aber wie eine Toolbox
  - Die gute Lösung muss engineered werden
- Gute Suche ist **nicht nur** mit Textvergleichen gemacht
  - Kann erst mit der Analyse des **Benutzerverhaltens** erreicht werden
- SearchKit - super Framework für einen Datenexplorer!
  - <http://searchkit.co/>
- Mit Kibana ein Super Tool für die Visualisierung von Timestamped-Daten



# Ressourcen

## Diese Präsentation & Demodaten:

<https://github.com/iterativ/elastic-presentation>

## Album-Explorer:

<https://github.com/iterativ/elastic-dataexplorer>

<http://data.iterativ.ch>

---

Datenquelle:

<https://musicbrainz.org/>

Album Explorer is using the SearchKit framework by:

<http://www.teneleven.co.uk/>

<http://searchkit.co/>

Elasticsearch & elastic:

<https://www.elastic.co/>

Elasticsearch is a trademark of Elasticsearch BV, registered in the U.S. and in other countries

# **Merci fürds zuelose!**



Pawel & Marco

[pawel.kowalski@iterativ.ch](mailto:pawel.kowalski@iterativ.ch)  
[marco@schess.ch](mailto:marco@schess.ch)