Universität Leipzig

Information Retrieval Laborprojekt

Evaluation der Geocache-Suchmaschine

Steven Lehmann, Fabian Ziegner, Christian Schlecht

supervised by
Jun.-Prof. Dr. Martin Potthast

23. September 2018

Inhaltsverzeichnis

1 Begriffserklärungen

1.1 Geocaching

Unter Geocaching wird eine Art Schnitzeljagd verstanden. Das Wort leitet sich vom griechischen $\gamma\widetilde{\eta}$, gē, stehend für "Erde" und dem englischen "cache" für "geheimes Lager" ab[1]. Ziel ist es, mit Hilfe von GPS-Koordinaten und/oder veröffentlichten Tipps einen versteckten Behälter - den sogenannten "Geocache" (auch kurz "Cache") - zu finden. In der

[1] https://de.wikipedia.org/wiki/Geocaching

Regel beinhaltet dieser Behälter ein Logbuch, in welches sich die Geocacher bei erfolgreicher Suche eintragen können. Zusätzlich, sofern es die Größe des Behälters hergibt, sind oftmals kleine Gegenstände zum Tausch vorhanden. Wichtig ist es, während der gesamten Zeit möglichst unentdeckt zu bleiben, sodass der Cache stets verborgen bleibt. Nach Beendigung der Suche wird der Cache wieder an seine ursprüngliche Position gebracht.

Es gibt eine ganze Reihe Arten von Geocaches. Von einfachen traditionellen Caches, über Multi-Caches und Rätsel-Caches bis hin zu virtuellen Caches können Suchen veranstaltet werden.

Fairness wird beim Geocaching groß geschrieben. Personen, die den Cache verändern, mutmaßlich zerstören, entfernen oder Dinge daraus stehlen wird größter Missmut entgegengestellt.

2 Motivation

2.1 Domäne

2.1.1 Geocaching als Sport

Geocaching erfreut sich zunehmender Beliebtheit unter der Bevölkerung, obwohl es in gewissem Sinne das Ziel ist, geheim zu agieren. Dieser Sport kann von Menschen aller Altersgruppen und körperlicher Verfassung betrieben werden, da in praktisch jeder Region (Wald, Stadt, Parkanlagen, Gebirge, ...) Caches versteckt und gefunden werden können. Unterbewusst wird dadurch die Gesundheit gefördert. Aus dieser Hinsicht ist der Sport in jedem Fall zu unterstützen.

2.1.2 Technischer Stand

Mittlerweile wird Geocaching durch das Internet stark unterstützt und vereinfacht. Während in den Anfängen (1980er Jahre) die Hinweise und Koordinaten meist verbal oder handschriftlich verbreitet wurden, gibt es heutzutage diverse Plattformen und Foren, um sich auszutauschen. Zu den bekanntesten gehören hier geocaching.com und opencaching.de im deutschsprachigen Raum. Diese Seiten agieren im Wesentlichen als Foren zum Austausch mit anderen Cachern. Sie bieten zwar die Möglichkeit der Suche, jedoch wird hierbei auf eine Datenbank zurückgegriffen, aus welcher mittels SQL-Anweisungen Suchanfragen bearbeitet werden. Wir haben keinen Anbieter feststellen können, der bei der Suche Methoden des Information Retrieval, d.h. Verwendung eines Index und Retrieval-Modells, anwendet. An dieser Stelle wird im Rahmen dieses Projekts prototypisch gearbeitet.

2.2 Dokumentkorpus

Die Geocaching Suchmaschine soll für Geocaches in Deutschland angewendet werden. Aus diesem Grund wurde als Dokumentkorpus eine Sammlung von allen [1] Geocaches in Deutschland, welche auf opencaching.de veröffentlicht sind, gewählt. Der Korpus beläuft sich damit auf eine Größe von ca. 20.000 Caches. Aufgrund des Mehraufwandes eines Web-Crawls wurden diese manuell heruntergeladen. Die Dokumentsammlung kann also als statisch betrachtet werden.

2.2.1 Dokumentstruktur

Jedes Dokument kann von opencaching.de in diversen verschiedenen Formaten heruntergeladen werden, darunter auch direkt von GPS-System verwertbare Formate. Aufgrund der Vollständigkeit der Information wurden in diesem Rahmen Caches im .txt-Format verwendet.

Das Dokument wird bereits beim Herunterladen in eine wohlgeformte Struktur gebracht. Die Abbildung 2.1 zeigt einen Ausschnitt. Es lässt sich eine Art Key-Value-Struktur erkennen, welche das spätere Parsing der Dokumente vereinfacht, sodass eine maximale Korrektheit gewährleistet werden kann. Interessant ist hierbei, dass durch dieses Dokument sämtliche Informationen aus der Web-Darstellung übernommen werden, die Informationen sind also a prior vollständig.

3 Architektur

3.1 Grobstruktur

Die Geocache Suchmaschine funktioniert nach dem Client-Server Prinzip. Als Programmiersprachen wurde für das Backend Java und für den Client natives HTML/CSS und JavaScript verwendet. Das Frontend empfängt angeforderte Daten vom Server (im Wesentlichen Beantwortung der Suchanfragen) über eine REST-Schnittstelle, welche mit Hilfe des Jersey - Frameworks [1] erstellt wurde.

3.2 Backend

Der Server wird als Webapplikation mit Hilfe einer Tomcat-Instanz als Container betrieben. Die Suchmaschine wurde mittels Apache Lucene [2] implementiert. Daneben existiert

[1] Stand 24.04.18
[1] https://jersey.github.io/
[2] http://lucene.apache.org/

```
Name: Unverhofftes Glück von Beeped Piano
Koordinaten: E 009° 27.066' N 49° 05.045'
Status: kann gesucht werden
Zustand: ok

Versteckt am: 01.11.2003
Wegpunkt: OC0ABC
Land: Deutschland
Cacheart: normaler Cache
Behälter: normal
D/T: 3.5/4.5
Online: https://www.opencaching.de/viewcache.php?wp=OC0ABC

Kurzbeschreibung: Am besten parkt man bei N  49° 05.077   E 009° 27.161

Beschreibung:
<====================>
Hallo Freunde!
Habe heute einen 2. Cache versteckt.
Vorab, das Gelände ist ein bisschen schwierig. Die Koords weichen unterschiedlich ab, mal 6 Meter mal 13Meter.
Zur Geschichte der Gegend erfährt man auf einer Hinweistafel am Parkplatz mehr.
Der Silberstollen ist nicht begehbar,dafür aber das Soldatenglück. Wer interesse hat hineinzugehen,sollte Gummistiefel und Taschenlampe nicht
So nun viel Spass am suchen.

Cacheinhalt: 1 Stein der Weissen
             1 Spiel die Siedler 2
             1 Regen Poncho
             1 Taschenlampenuhr
             1 lustiger Schlüsselanhänger
             2 mal eine Kinder Überraschung(nichts essbares)
             1Kugelschreiber sowie ein Bleistift
             1 Taschenrechner (solar)
und das wichtigste: 1 Log Buch

© Beeped Piano, Opencaching.de, CC BY-NC-ND, Stand: 24.04.2018;
alle Logeinträge © jeweiliger Autor
<====================>

Zusätzliche Hinweise:
<====================>

<====================>
A|B|C|D|E|F|G|H|I|J|K|L|M
N|O|P|Q|R|S|T|U|V|W|X|Y|Z

Logeinträge:
<====================>
A Meise / 06.04.2018 20:35 / gefunden
```

Abbildung 2.1: Dokumenstruktur eines Geocaches

```
public class Geocache {

    private String name;
    private String coordinates;
    private String status;
    private String condition;
    private String hiddenAt;
    private String waypoint;
    private String cacheType;
    private String caseType;
    private float difficulty;
    private float terrain;
    private String link;
    private String descriptionSnippet;
    private String description;
    private String tips;
    private List<Log> logs;
```

Abbildung 3.1: Java Objekt zur Repräsentation eines Caches

eine REST-Schnittstelle zur Kommunikation mit dem Frontend. Im Folgenden werden die einzelnen Bestandteile genauer erläutert.

3.3 Kommunikation

Über das Resource-Package des Servers werden Daten zum und vom Frontend gesendet. Es existieren GET-Requests zum Suchen (sowohl einfache als auch erweiterte Suche) und für das Empfangen von Suggestions. Ein weiterer POST-Request ermöglicht das Übertragen des Loggings an den Server. Die strikte Trennung zwischen Backend und Frontend ermöglicht beispielsweise das einfache Austauschen des Frontends oder die Anbindung an bereits bestehende Services.

3.4 Parsing der Dokumente

Bevor der Index für die Suchmaschine erstellt werden kann, müssen die vorhandenen .txt-Dateien in ein verarbeitbares Format gebracht werden. Aufgrund der Leichtgewichtigkeit und intuitiver Lesbarkeit wird das JSON-Format verwendet. Jedes Dokument wird zeilenweise in den Parser eingelesen. Anhand der bereits vorhandenen Key-Value-artigen Strukturen (siehe Abbildung 2.1) wird nach den Keys gesucht, diese entfernt, und die übrigen Values, d.h. die eigentlichen Daten, zu einem Java-Objekt zusammengefasst. Die Abbildung 3.1 zeigt dieses.

Zuletzt wird dieses Objekt mit Hilfe der org.json Library [3] in ein JSON Objekt überführt und schließlich in eine .json-Datei geschrieben. Der Name der Datei setzt sich aus dem Waypoint-Attribut des Geocaches zusammen, welches eine eindeutige Zuordnung er-

[3] https://github.com/stleary/JSON-java

möglicht. Dieses Attribut wird bereits durch die Rohdaten von opencaching.de geliefert. Die .txt - Rohdaten werden ab diesem Zeitpunkt nicht mehr benötigt.

3.5 Architektur der Suchmaschine

3.5.1 Index

Der Index wird anhand der zuvor erstellten json - Dateien konfiguriert. Jeder Geocache stellt hierbei ein eigenes Document im Index dar. Analog erhält jedes Attribut des Geocache-Java-Objekts ein eigenes Field. Lediglich auf die Attribute Tipps und Link wird verzichtet, da diese für den Index irrelevante Information enthalten (Tipps sind in Caesar-Chiffre verschlüsselt, Link ist ein HTML-Link auf opencaching.de). Es ist äußert unwahrscheinlich, dass derartige Suchqueries auftreten. Die übrigen Attribute werden mittels 3-Grammen indexiert. Dies ist besonders effizient für einen "Search-as-you-type"-Ansatz, da Edge-N-Gramme genau den Fluss der Eingabe widerspiegeln. Aus Zeitgründen wurde diese Methode leider nicht mehr umgesetzt.

3.5.2 Suche

Die Geocache-Suchmaschine ermöglicht zwei Arten der Suche: Zum einen eine standardmäßige Suche mittels einer Query, und desweiteren eine erweiterte Suchmaske, durch welche zusätzliche Suchparameter einbezogen werden können.

einfache Suche

Die einfache Suche ermöglicht die Eingabe eines Query-Strings, nach welchem der Index durchsucht werden soll. Es können sowohl die Lucene-Query-Syntax als auch natürlichsprachige Anfragen beantwortet werden. Als QueryParser wird ein `MultiFieldQueryParser` verwendet. Dieser sucht in allen Fields der Dokumente nach der Query. Zuvor werden durch einen GermanAnalyzer sowohl alle "normalen" Stoppworte der deutschen Sprache, als auch domänenspezifische Stoppworte, gefiltert. Diese umfassen die Worte "Geocache" und "Cache", da diese für die Semantik der Suche nicht von Bedeutung sind (nahezu alle Dokumente enthalten das Wort Geocache oder Cache, da diese das Thema sind). Würde man diese Worte nicht filtern, so würden Ergebnisse stark verfälscht werden (nähere Erläuterungen siehe Kapitel 4). Nach eingänglicher Betrachtung der Dokumente wurde sichtbar, dass zum einen der Name des Caches und zum anderen die Beschreibung die Felder sind, welche die meisten, d.h. die am such-relevantesten, Informationen enthalten. Aus diesem Grunde erhalten der Name einen Boost von 2.5 und die Beschreibung einen Boost von 1.5. Denkbar wäre auch ein Boost für die Logs zu etablieren, jedoch enthalten diese bei genauerer Betrachtung vielmals "nur" Vermerke über das Auffinden, sowie persönliche Kommentare zu Schönheit des Caches oder Tauschverhalten. Aus diesem Grunde werden die Logs nicht geboostet.

erweiterte Suche

Die erweiterte Suche bietet genau wie die einfache Suche die Möglichkeit zur Eingabe eines Query Strings. Zusätzlich dazu können Parameter, welche die Suche verfeinern, angegeben werden (z.B. nur Suche nach bestimmten Cache-Arten). Jeder Parameter repräsentiert dabei ein Field im Document. Für jeden angegebenen Parameter wird ein separater `SimpleQueryParser` verwendet, der im jeweilig korrespondierenden Field sucht. Die Query wird analog zur einfachen Suche verarbeitet. Im letzten Schritt werden nun die einzelnen Queries für die Parameter mit der Suchquery verschmolzen, um im Endeffekt nur eine einzige Suche durchführen zu müssen. Dies geschieht mit Hilfe eines `BooleanQuery.Builder`'s mit dem Junktor `BooleanOccur.MUST`.

3.5.3 Suggestions

Die Suchmaschine besitzt eine Autocompletion in etwas abgewandelter Form als gebräuchlich. Anstatt die Query beim Tippen Wort für Wort zu vervollständigen, werden stattdessen direkt Namen von Caches vorgeschlagen, welche zur aktuellen Eingabe passen. Da in der deutschen Sprache sowohl Prä - als auch Suffixe verbreitet sind, müssen diese geeignet erkannt werden, um falsche bzw. nicht ausreichend passende Vorschläge auszuschließen. Hierbei wird ein `AnalyzingInfixSuggester` verwendet, welcher von Lucene bereitgestellt wird. Da dieser Suggester einen eigenen Lookup-Index aufbaut, muss Redundanz in Kauf genommen werden. Dieser Index ist jedoch deutlich kleiner als der eigentliche Suchindex, da er lediglich die Namen der Caches indexiert und alle übrigen Informationen ignoriert. Das Ranking des AnalyzingInfixSuggester's erfordert eine Gewichtung. Diese wird anhand der Anzahl der Logs berechnet. Caches, die mehr Logs enthalten, sind stärker frequentiert und damit tendenziell relevanter als Andere mit weniger Logeinträgen. Somit werden diese als Erstes vorgeschlagen.

4 Evaluation

Zur Evaluation wurden insgesamt 14 Topics erstellt, die exemplarisch die Mehrheit der möglichen Suchanfragen abdecken sollen. Es handelt sich dabei um Stichwortsuchen sowohl mit als auch ohne erweiterte Suchparameter. Für jedes Topic wurden die besten 10 Suchergebnisse evaluiert. Das Relevanzmaß ist *Precision@k* und die daraus resultierende *Average Precision*. Tabelle 4.1 zeigt die verwendeten Suchquery mit ihren jeweils errechneten Precisions. Es ergibt sich letztendlich eine *Mean Average Precision* von 0.60.

Aus der Tabelle lassen sich eine Reihe von Schlüssen ziehen:

- Stichwortsuchen, welche durch den Analyzer auf einzelne Worte gefiltert werden (ID 1,6,8), erreichen perfekte Precision Werte. Zum Vergleich, bei einer Suche mit Topic 1, in welcher aber das Wort Geocache nicht als Stoppwort gefiltert wird, sind die

Tabelle 4.1: Topics zur Evaluation

ID	Suchquery	Kategorie	Average Precision
1	*Geocache im Wald*	Stichwortsuche	1.00
2	*sonniger Geocache*	Stichwortsuche	0.58
3	*Leipzig Auenwald*	Stichwortsuche	0.64
4	*Fahrrad erreichbar*	Stichwortsuche	0.62
5	*leicht zu finden*	Stichwortsuche	0.13
6	*Cache mit Aussicht*	Stichwortsuche	1.00
7	*See in Leipzig + cacheType=Multicache*	Stichwortsuche mit Parameter	0.5
8	*Erzgebirge*	Stichwortsuche	1.00
9	*Stadt + caseType=mikro*	Stichwortsuche mit Parameter	0.43
10	*Brandenburg + condition=ok + difficulty=2*	Stichwortsuche mit Parameter	0.64
11	*weit entfernt + cacheType=Multicache*	Stichwortsuche mit Parameter	0.23
12	*keine Hinweise*	Stichwortsuche	0.14
13	*Hund mitnehmen*	Stichwortsuche	0.57
14	*virtueller Cache nahe Berlin*	Stichwortsuche	0.93

Top-Ergebnisse genau diese, welche auf häufigsten das Wort "Geocache" enthalten. Diese haben nichts mit dem eigentlich wichtigeren Stichwort "Wald" zu tun. Bei derartigen Suchen wäre es interessant, die Bewertung auf ein k $>$ 10 anzuheben, da mit diesen Querys durchaus auch eine Suche, welche auf Vollständigkeit abzielt, durchgeführt werden könnte.

- generell erzielen Stichworte deutlich höhere Werte als Phrasen. Dies lässt sich einerseits darauf zurückführen, dass die Felder Name und Description Boosts enthalten, jedoch die Phrasen dort nicht explizit vorkommen, sondern sich beispielsweise in den Logs befinden. Desweiteren werden auch die Phrasen durch den QueryParser als Stichworte behandelt, so kann es passieren, dass die eigentliche Phrase nur sehr 'zerstückelt" (d.h. große Abstände zwischen den Wörtern, z.B. bei ID 5,11, 12) vorkommt, womit der Kontext eventuell ein ganz anderer ist. Ein Mechanismus zur Erkennung von Phrasen sollte hierbei Abhilfe schaffen können.
- Querys, welche Orte bzw. andere Named Entities enthalten, haben sehr große Chancen relevante Dokumente zu finden, weil Geocaches oftmals in der Nähe solcher versteckt sind, bzw. die Wegbeschreibung sich oft an solch markanten Punkten orientiert.

5 Ausblick

Um die Suchmaschine noch zu verbessern, könnten weiterblickend folgende Aspekte verfolgt werden:

- "Search-as-you-Type"-Ansatz: Suchergebnisse können direkt bei der Eingabe angezeigt werden. Da unser Index bereits auf 3-Grammen basiert, kann dieser auch dafür verwendet werden. Jedoch müsste die Performance der Suche verbessert werden.
- Query-Refinement: Mithilfe der gespeicherten Logs könnte die Query verändert werden, um bessere Ergebnisse zu erzielen.
- Erkennung von Phrasen für bessere Precision
- Verwendung von Fuzzy Querys, welche auf der Levenshtein-Editierdistanz beruhen. Damit können Tippfehler behandelt werden, um trotzdem passende Ergebnisse zu finden.