UNIVERSITÄT LEIPZIG

Information Retrieval - Praktikum

Wine Search

Timo Lehmann, Maik Bachmann, Martin Lorenz

beaufsichtigt von:

Junior-Prof. Dr. Martin Potthast

11-08-2019

Inhalt

1. Einleitung

Der folgende Bericht soll unser Vorgehen und unsere Entscheidungen für das Projekt Wine-Search erläutern. Wine-Search ist eine Suchmaschine, welche im Rahmen des Moduls *Information Retrieval* entwickelt wurde. Das Ziel des Projekts war es, an einer selbstgewählten Domäne (Weinrezensionen) die aus der Vorlesung vermittelten Inhalte selbst praktisch auszuprobieren und zu vertiefen. Die Schwerpunktsetzung war uns relativ freigestellt, aber die Anwendung sollte nach Aufgabenstellung auf einem Server laufen, mittels eines für Suchmaschinen üblichen Web-Interfaces bedienbar sein, einen Evaluierungsmodus besitzen und die standard Funktionsweise einer Suchmaschine haben. Letzteres bedeutet konkret, die Daten zu Indexieren, in diesen Daten zu suchen und alle Anfragen mit den dazugehörenden relevanten Informationen zu loggen.

2. Motivation

Das Thema Wein und insbesondere auch das Trinken von Wein beschäftigt die Menschen schon seit mehreren Tausenden von Jahren[1]. Im Laufe der Zeit wurden viele neue verschieden Rebsorten gezüchtet, welche heute abhängig von ihren ganz individuellen Anforderungen weltweit angebaut werden. Diese Vielfalt macht es einem schwer, den Überblick zu behalten bzw. einen neuen Wein für sich zu entdecken, der einem schmeckt. Allerdings gibt es Portale im Internet, auf denen anerkannte Wein-Rezensenten ihre Beurteilungen für die verschiedensten Rot- und Weissweine aus aller Welt veröffentlichen. Eine solche Seite ist beispielsweise WineEnthusiast, von welcher unser Datensatz stammt. Mittels Wine-Search kann dieser nun vom Nutzer durchsucht werden, wobei nicht nur Eigenschaften wie Farbe, Aroma, Rebsorte und Herkunftsland, sondern auch Eigenschaften wie der Preis berücksichtigt werden.

3. Der Datensatz

Der Datensatz mit den Wein-Rezensionen stammt von der Plattform Kaggle, wobei die Reviews von der Seite winemag.com gecrawlt wurden. Es liegen fast 130.000 Rezensionen im CSV-Format vor, wobei die neben der Rezension an sich noch folgende Felder für fast alle Rezensionen gegeben sind: Ursprungsland des Weins, die Kellerei, der Titel, welcher oft auch den Jahrgang enthält, der Preis und die Provinz bzw. Region aus der der Wein im Ursprungsland stammt. Es gibt auch Attribute, die weniger relevant sind. Dies wären der Tester bzw. der Rezensent und die von ihm subjektiv vergebene Anzahl an Punkten für einen Wein. Wobei letztere dennoch einen leichten Einfluss auf das Ranking haben können. Im folgenden wird darauf eingegangen, wie wir zu dieser Schlussfolgerung kamen.

3.1. Analyse des Datensatzes

Das Ziel der Analyse bestand darin, sich einen Überblick über den Datensatz zu verschaffen, um im Nachhinein Queries zu formulieren, die den Datensatz größtenteils abdecken. Es sollten also auch Themen gefunden werden, die nicht auf den ersten Blick offensichtlich waren.
Dazu wurde im ersten Schritt der Datensatz „per Hand“ analysiert, d.h. es wurden Auswertungsmöglichkeiten genutzt, die einem ein Standard Tabellenprogramm bietet. Ein Beispiel dafür ist die Anzahl unterschiedlicher Werte. So wurde mittels dieser Vorgehensweise ermittelt, dass die 130.000 Rezensionen von nur 19 verschiedene Rezensenten erstellt wurden und dass fast 20% aller Rezensionen auf eine Person zurückgehen. Da somit fast 26.000 Rezensionen von einer Person stammen, ist das Suchen nach Rezensionen von bestimmten Personen eher irrelevant.
Des Weiteren viel auf, dass es zwar eine subjektive Anzahl vergebener Punkte auf einer Skala von 0 bis 100 gab, dass nur Punkte im Bereich von 80 bis 100 vergeben wurden. Wobei sich hier 90% der Rezensionen im Bereich von 84 bis 94 Punkten bewegen.
Ein weiterer Punkt, der uns zu einer tiefgründigeren Analyse veranlasste war, dass die Rezensionen alle recht ähnlich waren. Der Großteil beschreibt Geschmacksnuancen mit sehr

ähnlichen Wörtern, was natürlich ein konkretes Suchen erschwert. Ein weiterer auffallender Punkt war, dass sehr viele Adjektive verwendet wurden. Deshalb soll der nächste Teil dieses Kapitels beschreiben, wie mittels Methoden des Information Retrieval der Datensatz analysiert wurde.

3.2. Analyse der Rezensionen

Da hauptsächlich die Rezensionen durchsucht werden sollen, werden diese hier genauer betrachtet. Zu allererst wurde die durchschnittliche Rezensionslänge bestimmt. Dies wurde während des Indexierungsprozesses erledigt. Das Ergebnis war eine durchschnittliche Länge von 40 Wort, was für eine Volltextsuche recht kurz ist, aber noch toleriert werden kann.

3.2.1. Verwendete Adjektive

Da uns die Analyse bei der Formulierung der Queries helfen soll, wurde im nächsten Schritt die Wortwahl genauer betrachtet, konkret, die Häufigkeit der verwendeten Adjektive. Dies wurde mittels Apache OpenNLP und Apache Lucene realisiert. Dafür wurde ein Modul erstellt, welches mittels Part-Of-Speech-Tagging alle verschiedenen Adjektive zählt. Das Ergebnis dabei war, dass es ca. 11.000 verschiedene Adjektive gibt. Wobei ungefähr die Hälfte nur ein einziges mal vorkommt und ca. 1% mehr als 1000 mal vorkommen.

Wort	Häufigkeit
ripe	24975
black	24325
red	17177
fresh	16909
rich	16399
dry	13999
soft	12196
white	11993
bright	10870
dark	10713

Wort	Häufigkeit
good	9404
green	8585
crisp	8259
smooth	6523
fine	5843
creamy	5605
tannic	5522
clean	5486
juicy	5435
full-bodied	5222

Wort	Häufigkeit
dense	5002
ready	4984
great	4934
elegant	4912
earthy	4653
savory	4652
tight	4512
delicious	4507
complex	4434
concentrated	4243

Abb 1: Die 30 häufigsten Adjektive

Für die Query-Formulierung sind nicht alle Adjektive relevant. Man wird wohl kaum eine Anfrage formulieren, die explizit Wörter wie *ripe, good, fine* oder *delicious* enthält. Dies sind eher implizierte Vorraussetzungen des Suchenden. Für Anfragen sind eher Worte wie *dry, red* oder *white* interessant, allerdings stellt dies keinen neuen Erkenntnis gewinn dar. Worte wie soft, *smooth, creamy, full-bodied* oder *concentrated*, sind wohl eher der begeisterten Ausdrucksweise des Rezensenten zuzuschreiben, als das es wirklich relevante Suchbegriffe wären.
Dennoch wurde mit dieser Auswertung und etwas Suchaufwand das Themengebiet „Bio“ herausgearbeitet und enthält Wörter wie *organic, biodynamic, regional, demeter-certified* und *certified-organic*. Dies ist jetzt eins von potentiell vielen Themen.

3.2.2. Clustering mittels LSI

Um einen besseren Überblick über die verschiedenen Themen der Rezensionen zu bekommen, wurde mittels des LSI-Verfahrens eine Übersicht der Themen anhand der verwendeten Wörter erstellt. Dafür wurde *Python Gensim* verwendet. Bei einem ersten Durchgang erhielten wir das Ergebnis: 4 verschiedene Themen bei einer maximalen Kohärenz von 0,468. Man hat also selbst bei vier großen Themenkomplexen noch eine sehr hohe Überschneidung, was auch durch *Abb. 1*

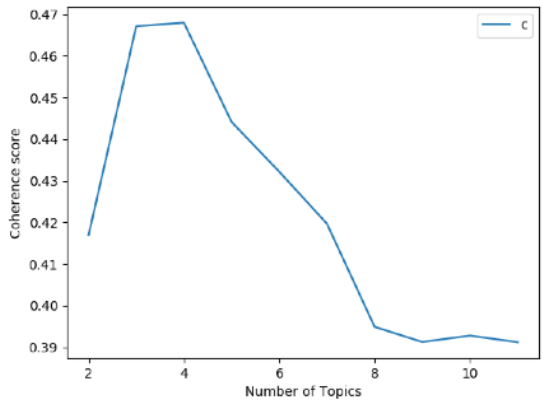

Abb. 2: *Kohärenz Wert in Abhängigkeit der Themenanzahl bei Durchgang 1*

1: wine, flavor, fruit, aroma, acid, finish, palate, drink, tannin, cherry, black, ripe, dry, spice, note, rich, red, fresh, berri, show

2: wine, aroma, palat, finish, flavor, cherri, note, fruit, age, nose, black, drink, rich, berri, plum, spice, offer, acid, fruiti, ripe

3: black, tannin, flavor, cherri, appl, acid, finish, fresh, citru, lemon, crisp, peach, pear, white, dark, red, spice, cabernet, blackberri, green

4: fruit, flavor, palat, wine, cherri, fresh, blackberri, white, note, acid, nose, app, offer, oak, citru, cabernet, alongsid, sweet, chocol, hint

Abb. 3: *Häufigste Wortstämme in den Themen und ihre Titel, 1: fruchtige trockene Rotweine, 2: würzige Rotweine mit pikanter Säure, 3: Rezensionen mit vielen Geschmacksnuancen, 4: fruchtige Weißweine*

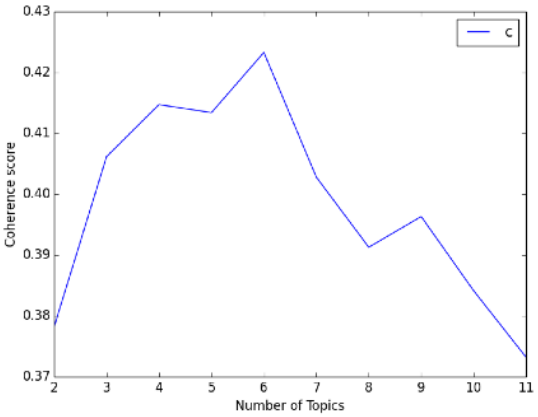

Abb. 4: *Kohärenz Wert in Abhängigkeit der Themenanzahl bei Durchgang 2*

1: fruit, flavor, aroma, finish, acid, drink, tannin, berri, oak, note, plum, show, full, nose, blackberri, offer, blend, sweet, age, appl

2: fruit, flavor, finish, aroma, drink, age, acid, tannin, plum, berri, give, nose, feel, note, wood, appl, well, charact, structur, year

3: tannin, flavor, aroma, acid, berri, appl, cabernet, plum, blackberri, fruit, lemon, offer, peach, pepper, alongsid, blend, drink, licoric, lime, sweet

4: fruit, drink, acid, finish, aroma, tannin, flavor, fruiti, age, give, year, textur, note, well, blackberri, nose, charact, feel, full, structur

5: flavor, acid, aroma, appl, finish, offer, cabernet, lemon, blackberri, nose, note, fruit, peach, hint, alongsid, miner, oak, blend, flower, lime

6: aroma, finish, cabernet, note, blend, flavor, berri, sauvignon, oak, nose, merlot, show, fruit, blackberri, franc, lemon, petit, sweet, appl, acid

Abb. 5: *Häufigste Wortstämme in den Themen bei Durchgang 2 und ihre Titel,*
1: beerige Weine, 2: trinke jetzt / ab Jahr 3: tanninreiche Weine mit leichter Säure aus Rebsorte Cabernet 4: Textur hat noch Potenzial, deshalb noch bis zu bestimmtem Jahr liegen lassen 5: Wein aus dem Eichenfass 6: Cabernet Sauvignon und C. Franc

bestätigt wird. Also wurden im zweiten Schritt die häufigsten Wörter wie *wine, palate* und auch die 120 häufigsten Adjektive entfernt.
Im zweiten Durchgang sah das Ergebnis etwas anders aus. Jetzt sind sechs Themenbereiche erkennbar, welche sich allerdings noch stärker als in Durchlauf 1 überschneiden, was auch an einem Kohärenzwert von 0,422 erkennbar ist. Des Weiteren wurde das LSI-Verfahren auf die Wortstämme angewandt, so dass wirklich Themen erkennbar sind. Allerdings erschweren diese durch den Porter-Stemmer erstellten Stämme etwas die Auswertung. Dennoch wurde versucht, sinnvolle Themenkategorien (*Abb. 5*) zu finden, tiefere Einblicke in den Datensatz bzw. das Erkennen von ganz neuen Themen wie dem *Bio*-Thema sind allerdings ausgeblieben.
Trotzdem stellen beide Auswertungen einen guten Ausgangspunkt für die Query-Formulierung dar, da Themenkomplexe erkennbar sind, auf welche mit gezielter Query-Formulierung zugegriffen werden kann. Gut geeignet erscheinen *fruchtige Weißweine, fruchtige trockene Rotweine, würzige Rotweine* oder auch *Weine zum jetzt / später trinken*.

4. Technologien

4.1. Apache Lucene

Als Framework für das Backend unserer Suchmaschine nutzen wir das auf Java basierende Apache Lucene. Dabei handelt es sich um eine von der Apache Software Foundation kostenfrei zur Verfügung gestellte Programmbibliothek, welche unter Anderem von Webseiten wie Twitter verwendet wird. Lucene bietet die Möglichkeit einen Index aus beliebigen Daten aufzubauen. Dabei wird der von Stoppwörtern befreite Text indiziert.
Als Retrievalmodell wird das Vektorraummodell aber auch ein tf-idf-Maß eingesetzt.
Des Weiteren stellt das Framework Methoden, zur Verfügung welche das Ranking aber auch die Queryverarbeitung erleichern.

4.2. Apache Tomcat

Apache Tomcat ist ein Open-Source Web-Server, welcher sehr viele Java EE Spezifikationen implementiert. Dies wären beispielsweise Java-Servlets oder auch JavaServer Pages (JSP), auf welchen unsere Anwendung basiert.

5. Aufbau und Funktion

Dieser Abschnitt beschreibt den allgemeinen Aufbau und die Funktion unserer Weinsuche. Er enthält Informationen zur Suche an sich, unserer Modifizierung der eingegebenen Queries und zum Evaluationsmodus.

5.1. Allgemeine Funktionsweise

Die Suchmaschine besteht im Kern wie alle Suchmaschinen aus einem Indexer und einer Suchfunktion. Zusätzlich besitzt sie ein Web Interface, mit dessen Hilfe der Nutzer die Weinsuche bedienen kann.
Der Indexer erstellt den Index, auf welchem die Suchfunktion nach den eingegebenen Queries sucht. Die Ergebnisse werden mithilfe einer jsp-Datei im Browser angezeigt und können dort auf Wunsch auch bewertet werden. Die Verarbeitung der Anfragen geschieht über die zwei Servlets, dem Query- und dem Evaluation-Servlet.
Sämtliche Anfragen werden mit IP-Adresse und Timestamp geloggt.

5.2. Der Indexer

Der von uns verwendete Indexer ist im Prinzip der Standardindexer, der von Lucene bereitgestellt wird (IndexFiles.java). Wir haben ihn modifiziert um nicht nur Doc-Dateien, sondern auch CSV-Dateien indexieren zu können. Dies hat den Vorteil, das wir die gegebene Struktur unseres Datensatzes ausnutzen können um die Daten in entsprechende Felder zu indexieren.
Die Daten werden aufgeteilt in die Felder ID, Country, Description, Designation, Points, Price, Province, Taster, Title, Variety und Winery.
ID ist hierbei einfach nur eine individuelle Identifikationsnummer der einzelnen Reviews, Title die Überschrift und Description den Text der Reviews. Dies sind die wichtigsten Felder für unsere Suche.
Country bezeichnet das Herkunftsland des Weins, während Winery das Weingut in ebendiesem Land enthält. Province bezeichnet den Staat oder die Provinz in dem das Weingut liegt. Als Ergänzung dazu enthält Designation auch noch den Abschnitt des Weinguts, von dem die Trauben für den Wein stammen. Bis auf das Herkunftsland waren diese Angaben jedoch wenig interessant für uns. Variety bezeichnet die Sorte der Trauben.
Points enthält die Bewertung des Weins, Price den Preis und Taster den Tester. Besonders Points und Price sind für die Sortierung der Ergebnisse wichtig, davon abgesehen jedoch von keinem besonderen Nutzen.

Der Indexer speichert die indexierten Daten in einem Ordner namens Index im aktiven Programmordner des Programms ab. In diesem Programmordner muss auch die CSV-Datei mit den Rezensionen liegen.

5.3. Die Suche

5.3.1. Der Searcher

Für die Suche verwenden wir eine eigene Suchfunktion in der Klasse QueryServlet.java. Die bereitgestellte Klasse IndexReader liest die Ergebnisse aus dem Index für eine Query aus. Die Query selbst wird vom StandardAnalyzer verarbeitet. Lucene bietet eine absolute Suche, d.h. Schreibfehler und ähnliches werden mit berücksichtigt. So würde „wine“ beispielsweise sehr häufig gefunden werden, „winet“ jedoch gar nicht.
Standardmäßig haben wir einen MultiFieldQueryParser eingebaut. Dieser durchsucht alle Felder. Zur besseren Übersicht werden allerdings nur Title, ID, Description, Price, Points und Country angezeigt.
Lucene gibt uns anschließend die besten 500 Ergebnisse in einer Liste zurück. Diese werden mithilfe der scoreDocs() Funktion bewertet. Da unser Datensatz sehr homogen ist und dementsprechend viele Ergebnisse liefert, die aller sehr ähnlich sind, haben wir beschlossen, aus dieser bewerteten Ergebnismenge nur die Top 50 anzuzeigen. Aus diesem Grund werden die Top50 Reviews in der resultsList gelistet und zur weiteren Verarbeitung genutzt.
Für die verbesserte Suche haben wir uns vor allem mit QueryExpansion beschäftigt. Diese läuft allerdings nur im Evaluationsmodus,damit ein vorher - nachher Vergleich möglich ist.

5.3.2. Die Verarbeitung der Queries

Um die Suche zu verbessern und die Ergebnisanzeige zu optimieren, haben wir uns hauptsächlich mit der Vorverarbeitung der Queries beschäftigt. Ziel dabei war es, die Queries so zu erweitern, dass relevantere Ergebnisse gefunden werden. Ein Problem war die absolute Suche von Lucene. Nehmen wir als Beispiel den Suchstring „dry red“. Lucene durchsucht mit diesem String alle Dokumente nach „dry“ und dann nach „red“. Das Ergebnis ist eine Mischung aus beiden Ergebnismengen. Die Query sollte eigentlich trockenen Rotwein finden, jedoch wird in den Ergebnissen auch trockener Weißwein angeboten. Um solche Fälle zu vermeiden, wurden mehrere Prüffunktionen implementiert, die die Query erweitern oder trimmen, um präziseres Suchen zu ermöglichen. Deshalb wird im folgenden auf einzelne Funktionen eingegangen.

5.3.2.1. check()

Die check() Funktion dient als Sammelpunkt aller Prüffunktionen, hier können einzelne Überprüfungen abgeschaltet werden und leicht neue Prüffunktionen hinzugefügt werden. Sie gibt zur Kontrolle außerdem den alten und den neuen Suchstring in der Konsole aus.

5.3.2.2. checkfrom()

CheckFrom() prüft ob ein String die Formulieren „from XXXX“ enthält. Wir nehmen an, dass Nutzer damit nach Weinen aus dem angegebenen Jahr suchen und erweitern die Query deshalb mit dem Keyword „vintage“.

5.3.2.3. checkvintage()

Diese Funktion reagiert auf das Keyword „vintage“. Ist dieses Wort vorhanden, prüft sie ob eine vierstellige Zahl in der Query gegebenen ist und schreibt „title:“ davor, allerdings nur, wenn es eine valide Jahreszahl zwischen 1900 und 2100 ist. Da Weine häufig ihren Jahrgang im Titel haben, werden so präzisere Ergebnisse gefunden.
Hierbei muss gesagt werden, dass Lucene diese Aufgabe standardmäßig besser erfüllt, wenn nur die Jahreszahl eingegeben wird. Aufgrund potenzieller weiterer Angaben im Suchstring wurde jedoch diese Herangehensweise gewählt. Das Auftreten im Titel wurde auch nicht erzwungen, damit auch Jahrgänge, die nur wenige Jahre entfernt von dem gesuchten sind, ebenfalls ausgegeben werden.

5.3.2.4. checkrecommend()

Diese Funktion erweitert die Query stark und sucht nach bekannten Empfehlungsformulierungen wie „drink now“, „drink in ...“ und ähnlichen Ausdrücken. Auch hier müssen diese Ausdrücke nicht im Text vorkommen, werden jedoch mitgesucht.

5.3.2.5. checktype()

Checktype prüft mithilfe einer Prüfgröße ob Rot- oder Weißwein gesucht wird. Je nachdem welcher Typ gesucht wird, erweitert die Funktion die Query um „-blanc -white“ oder “-red -rouge“. Dies liefert bereits viel präzisere Ergebnisse bei Queries wie „dry red“, findet aber noch Weißweine, bei denen die entsprechenden Keyworte nicht verwendet wurden (häufig in Riesling Reviews). Hierbei würde dann die Evaluationsfunktion zum Einsatz kommen.

5.3.2.6. checkcountry()

Jeder Wein hat nur ein Herkunftsland. Diesen Umstand machen wir uns zunutze, indem wir Landangaben, die in der Query auftauchen, parsen und Lucene mit „+country:...“ dazu zwingen, das Feld Country nach genau diesem Herkunftsland zu durchsuchen. So finden wir mit der Query „german wine“ auch nur Weine aus Deutschland, was eine massive Verbesserung der Ergebnisse gegenüber der Standardsuche darstellt.

5.3.2.7. checkfood()

Gelegentlich enthalten die Rezensionen Empfehlungen zu welchen Mahlzeiten ein Wein passen würde. Die Funktion checkfood() prüft ob und zu welchem Essen Wein gesucht wird und sucht nach Beschreibungen, die explizit dieses enthalten.

5.3.2.8. checktaste()

Checktaste() funktioniert genau wie checkfood(), nur mit Geschmäckern wie „sweet“.

5.3.2.9. checkprice()

Checkprice() prüft nur, ob teurer oder billiger Wein gesucht wird und setzt eine Kenngröße für die spätere Sortierung. Dazu wird die Query auf die Keyworte „cheap“, „expensive“ und „pricey“ geprüft.

5.3.2.10. checkbest()

Falls Wein von hoher oder niedriger Qualität gesucht wird, so prüft diese Funktion das und setzt ebenfalls eine Kenngröße. Keyworte sind hier „good“, „best“, „bad“ und „worst“.

5.3.2.11. checkbio()

Checkbio() ist eine kleine Funktion, die in der Query „bio“ durch „organic“ ersetzt um dem Scoring entgegenzukommen.

5.3.2.12. Die Sortierung der Ergebnisse

Die Sortierung der Ergebnisse muss im Evaluationsmodus durchgeführt werden, um relevante Ergebnisse oben und irrelvante unten anzuzeigen. Aber auch anhand von Preis und Qualität kann sortiert werden.

5.3.2.13. sortcheap()/sortpricey()

Diese beiden Sortierfunktionen ordnen die 50 Top Ergebnisse für eine Query nach Preis, abhängig davon ob teure oder billige Weine gesucht werden. Zu diesem Zweck liest sie die gesetzte Kenngröße von checkprice() aus und sortiert die Ergebnisse nach Preis.
Die Sortierung selbst ist ein Bubblesort. Er wurde aufgrund seiner Stabilität und einfachen Implementation gewählt und liefert für die 50 Ergebnisse zufriedenstellende Laufzeiten.

5.3.2.14. sortbest()/sortworst()

Nach dem gleichen Prinzip wie die Preissortierung arbeitet auch die Qualitätssortierung. Der Unterschied ist, dass die Qualität gegenüber dem Preis als zweitrangig eingestuft wurde. Das bedeutet wenn wir nach dem besten billigen Wein sortieren, so werden Weine mit dem geringsten Preis ganz oben angezeigt, auch wenn ihre Qualität niedriger ist als nur leicht teurere Weine.
Die zugrundeliegende Annahme ist, dass für Nutzer die Kosten eines Weins ausschlaggebend für einen Kauf sind und nicht die Qualität. Sind also beide Sortierungen gefragt, so sind die Ergebnisse nach Preisklassen sortiert und innerhalb dieser nach Qualität. Dies wird durch die Stabilität des Bubblesorts ermöglicht.

5.3.3. Scoring

Scoring war kein Fokus von uns, weshalb es nur minimal vertreten ist. Jedoch werden die Worte „wine“ bzw „wines“, „fine“, „delicious“ und „vintage“ abgerankt, da sie häufig auftreten und wenig hilfreich sind.
„Organic“ wird dagegen geboostet, wenn es vertreten ist, da erfahrungsgemäß Nutzer, die auf so etwas achten, dies meist als essentiell betrachten.
Alle anderen Worte werden normal gescored.

5.4. Der Evaluierungsmodus

Der Evaluierungsmodus dient dazu, die von der Suche gefundenen Dokumente nach ihrer Relevanz zu bewerten. Er wird mittels des Häckchens auf der Startseite gesetzt.
Diese Information wird an das Queryservlet (QS) weitergegeben, welches veranlasst, dass für jedes Ergebnis auf der Suchergebnis-Seite eine Segmented-Control-Anzeige mit den Zuständen *Relevant, Nicht bewertet* und *Irrelevant* angezeigt wird. Standardmäßig ist der Wert auf nicht bewertet gesetzt. Sollte allerdings schon eine Bewertung für diese Query vorliegen, wird diese aus der entsprechenden JSON Datei ausgelesen und die Information wird für die Darstellung der Zustände mit an den Client übermittelt. Des Weiteren erfolgt hier auch das weiter oben beschriebene Ranking.
Client-seitig besteht nun die Möglichkeit, die verschiedenen Queries nach ihrer Relevanz zu bewerten. Dabei werden alle aktuelle Zustände, alle 5 Sekunden automatisch und asynchron an das Evaluierungsservlet (ES) übermittelt, welche die Daten nach einem vorgegebenen Schema in eine von der Query abhängige JSON-Datei schreibt. Auf diese Datei greift wiederum das QueryServlet zurück, um die Evaluationsdaten bei der nächsten Anfrage mit dieser Query an den Client zu übermitteln.

6. Evaluation

6.1. Durchführung

6.1.1.Methode

Die Evaluation lief folgendermaßen ab: Zuerst wurde der Evaluierungsmodus benutzt um durch alle topics durchzugehen und diese zu bewerten. Die Ergebnisse wurden anschließend sortiert und die Relevanz dokumentiert. Anschließend wurde das gleiche in der Standardsuche gemacht.
Als Bewertungsverfahren wurde (aufgrund von Zeitdruck) die precision für jede topic in beiden Suchen berechnet und anschließend gegenübergestellt.
Doppelte Reviews wurden dabei als irrelevant gekennzeichnet, da sie bereits aufgetaucht sind.

6.2. Auswertung

Wie die Gegenüberstellung im Anhang zeigt, liefert unsere modifizierte Suchmaschine in vielen topics bessere Ergebnisse als die Standardsuche bzw. zeigt überhaupt Ergebnisse an. Topics wie „*recommendations*“ funktionieren in der Standardsuche nicht, da keine der Reviews das Wort Recommendations enthält und Lucene deshalb nichts findet. Dies ist natürlich ein Vorteil für unseren Evaluierungsmodus.
Dennoch hat auch die Standardsuche ihre Stärken, beispielsweise bei dem topic „pricey red wine“. Dort ist die Suche dreimal so gut wie der Evaluierungsmodus. Dies liegt vermutlich daran, dass „wine“ im Evaluierungsmodus abgerankt wird. Die Standardsuche rankt also

Dokumente die „red“ und „wine“ enthalten viel höher als unsere modifizierte Suche, die größtenteils nur „red“ hoch rankt. Dadurch finden wir weniger Ergebnisse.
Insgesamt jedoch sind wir im Durchschnitt deutlich besser, besonders auch, da der Evaluierungsmodus die als relevant markierten Ergebnisse beim nächsten mal automatisch nach oben sortiert, wodurch eine viel höhere Präzision innerhalb der ersten Ergebnisse erzielt wird.

7. Ausblick

Im Vergleich zu anderen Suchmaschinen, könnte man zu dem Schluss kommen, dass unsere Suchmaschine noch etwas unvollständig ist. Dies ist aber dem Umstand geschuldet, dass auch ein nicht unerheblicher Teil der Entwicklungszeit in die Analyse des Datensatzes und die Entwicklung entsprechender Analyse-Module geflossen ist. Dennoch könnten wir uns für die Zukunft vorstellen, dass noch folgende Punkte verbessert und erweitert werden.
Zuallererst wäre da der Quellcode, welcher sicherlich noch strukturierter werden könnte. Darunter fällt beispielsweise die Auslagerung der check-Funktionen in eine eigene Java Klasse, um das QeryServlet nicht zu unübersichtlich zu machen. Auch die Sortierfunktionen könnten dort mit hinein oder ebenfalls in eine eigene Klasse. Eine Codekomprimierung täte dem Projekt ebenfalls gut, sowie eine bessere Ausnutzung bereits vorhandener, verarbeiteter Daten.
Des Weiteren könnten viele kleine Quality of Life Verbesserungen gemacht werden. Ein Beispiel dafür wäre eine visuelle Rückmeldung, wenn die Indexierung fertig ist oder ein Zurück Button auf der Ergebnisseite, so dass nicht immer die entsprechende Browserfunktion benutzt werden muss.
Obwohl der der Datensatz gründlich untersucht wurde, gibt es dennoch einige doppelte Reviews mit unterschiedlichen IDs die zu teils doppelten Ergebnissen führen. Des Weiteren stellte die Homogenität des Datensatzes eine Herausforderung dar, hierbei könnte die Indexierung zusätzlicher Daten von anderen Seiten helfen.
Eine tiefgreifenderere Queryexpansion ist natürlich ebenfalls möglich. Besonders bei der Unterscheidung zwischen Rot- und Weißwein wäre eine Voreinteilung nach Sorten nützlich, anhand derer die Ergebnisse gefiltert werden können. Beispielsweise wird bei der Query „*dry red wine*“ häufig Riesling vorgeschlagen. Dieser entgeht der checktype() Funktion, da die Sorte so bekannt ist, dass eine erneute Erwähnung, dass es sich hierbei um eine Weißweinsorte handelt, nicht stattfindet.
Für die Zukunft kann also gesagt werden, dass nicht hauptsächlich die Funktionalität, sondern eher die GUI und auch etwas die User-Experience verbessert werden kann.

8. Anhang

8.1. Gegenüberstellung der Topics

Für jedes einzelne Topic wurde die *precision* ausgerechnet und hier gegenübergestellt.

Topics	Standardsuche	Evaluierungsmodus
"best white wines"	0,86	0,96
"best red wines"	1	0,98
"best wines 2019"	0,36	0,92
"best wines from 2005"	0,46	0,96
"wine for meat dishes"	0,52	0,86
"wine for seafood"	0,44	0,94
"oak barrel wine"	0,84	0,72
"sweet red wine"	0,48	0,46
"dry red wine"	0,52	0,5
"fruity white wine"	0,5	0,66
"berry flavoured white wines"	0,02	0,14
"weak wines"	0,54	0,48
"strong wines"	0	0,58
"recommendations "	0	0,4
"french wines"	0,26	0,3
"italian wines"	0,72	0,96
"spanish wines"	0,48	0,74
"wine from germany"	0,5	0,94
"american wines"	0,94	0,98
"cherry wine"	0,48	0,58
"cabernet sauvigion cheap"	0,16	0,18
"good wine for more than 20$"	0,6	0,78
"semi-dry wine"	0,94	0,96
"riesling under 20$"	0,18	0,22
"spanish tempranillo"	0,4	0,74
"wine to drink while eating clams"	0	0,28
"wine that goes well with cheese"	0,06	0,18

"citrus note"	0,64	0,66
"smooth wines"	0,8	0,74
"wines for pasta"	0,52	1
"red organic"	0,34	0,54
"cheap organic"	0,14	0,14
"smooth fruity wines"	0,24	0,76
"pricey red wines"	0,74	0,24
"everyday wine"	0,84	0,84
"dry rosé"	0,8	0,86
Durchschnitt	~ 0,481	~ 0,644

9. Quellen

1 Patrick McGovern et al.: *„Early Neolithic wine of Georgia in the South Caucasus“*