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.
Sämtliche Anfrage an die Suche werden zusammen mit dem Datum, der Zeit und der Hostadresse gespeichert.

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. In diesem Programmordner muss auch die CSV mit den Informationen liegen.

5.3 Die Suche

5.3.1 Der Searcher

Für die Suche verwenden wir eine eigene Suchfunktion in der Klasse QueryServlet.java, die sich an der SearchFiles.java orientiert. 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, es werden also alle Felder durchsucht, obwohl letztendlich nur Title, ID, Description, Price, Points und Country angezeigt werden.
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, 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 nur im Evaluationsmodus, daher kann im normalen Modus die Standardsuche von Lucene ausgeführt werden.

5.3.2 Die Verarbeitung der Queries

Um die Suche zu verbessern und die Ergebnisanzeige zu optimieren, haben wir uns hauptsächlich mit der Verarbeitung 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.

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, falls 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“.

5.3.3 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.

5.3.3.1 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.

5.3.3.2 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.