Web-Genre Identifikation von HTML-Dokumenten

Autor: Matthias Rebel, 731220
rebel@@@uni-potsdamPUNKTde

Universität Potsdam
Seminar: Automatische Textanalyse
Dozent: Prof. Dr. Manfred Stede

Information
In dieser Arbeit wird ein Programmierprojekt zur Web-Genre Identifikation vorgestellt. Ziel ist es unbekannte HTML-Dokumente einem Genre zuzuordnen und somit die Suche im Internet zu verbessern, oder auch den Erstellungsprozess von Webkorpora zu unterstützen.
Bei der Implementierung wurde berücksichtigt, dass es keine 100%igen Festlegungen auf die Genres im Web gibt. Daher wird ein Genre lediglich durch eine vorher von Hand ausgewählte Menge an Dokumenten beschrieben. Eine Erweiterung um neue Genres oder eine Veränderung der Klassifikation ist durch den Aufbau sehr einfach zu bewerkstelligen.

Inhaltsverzeichnis

1 Einleitung

Der Begriff Genre stammt aus der bildenden Kunst. Er ermöglicht das Werk einer bestimmten Gattung, nach thematischem Inhalt zuzuordnen. Dies geschieht zum Einen durch die Zuordnung zu verschiedenen Gattungen, wie z.B. der Malerei, und den dazugehörigen Genres, wie Stillleben, Portrait, Landschaft usw.
In die Linguistik übernommen, wird dieser Begriff dazu verwandt, um Texte unter einem bestimmten Oberbegriff zusammen zu fassen, somit zu ordnen und durch Erkennen von Unterschieden von einander abzugrenzen. Einen genauen Einblick in die Entstehung des Genrebegriffs und seiner Entwicklung bis zum heutigen Zeitpunkt gibt die Dissertation von Rehm (2006).
Aufgrund der im Internet vorliegenden Billionen von mehr oder weniger ungeordneten Dokumenten ist auch hier eine Strukturierung in Form von Genres wünschenswert. Als Vorläufer zu dieser Arbeit wurde eine orientierende Umfrage unter Studierenden und Mitarbeitern der Universität Potsdam durchgeführt. Diese ergab, dass die Befragten es als sinnvoll erachteten, die im Web vorliegenden Dokumente zu klassifizieren, die Suche dadurch zu vereinfachen und effektiver zu gestalten.
Unter dem Thema der so genannten Webgenre-Klassifikation/Identifikation wird in dieser Arbeit ein Programm beschrieben, das ein im Internet vorliegendes HTML[1]-Dokument einem vorher festgelegten Genre zuordnet. Eine Vielzahl anderer Autoren beschäftigten sich bereits mit diesem Thema (Rehm (2006), Stein und zu Eissen (2008), Lahn (2006), Boese und Howe (2005)), um gegebenenfalls die erwünschte Benutzerfreundlichkeit in naher Zukunft gewährleisten zu können. Vergleichbar mit der Anzahl der vorliegenden Arbeiten ist die Anzahl der verschiedenen Webgenrelisten. Daher wird ein flexibles System benötigt, dass universal auf alle Genrearten angewandt werden kann. Obwohl in dieser Arbeit lediglich vier Genres berücksichtigt wurden, ist das System so konzipiert, dass es um beliebig viele Genres erweitert werden kann, ohne das System verändern zu müssen. Denn aufgrund wachsender Informationsmenge im Internet ist zudem eine Entstehung weiterer Genres nicht auszuschließen.

[1] Hyper Text Markup Language

2 Die Implementierung

2.1 Programmablauf

Zu Beginn wurde für jedes Genre von Hand eine Liste mit 50 URLs[2] erstellt. Das System erweitert diese automatisch auf 300 URLs und lädt von jeder URL den HTML-Code herunter und speichert ihn zusammen mit allen anderen Codes des Genres in einer Datei (Abschnitt 2.2). Folgend wird diese Datei verwendet, um ein genrespezifisches Wörterbuch zu generieren (Abschnitt 2.3.2). Hiermit ist die Initialisierung des Systems abgeschlossen.
Anschließend erfolgt das Einlesen der Datei, die den HTML-Code enthält, und die Weiterleitung jedes Dokuments durch eine Pipe in der sukzessive alle Merkmale extrahiert werden. Diese Merkmale werden in Form eines Vektors in einer Datei gespeichert, die alle Vektoren aller Genres enthält. Dadurch entstehen bei vier Genres von je 300 Dokumenten 1200 Vektoren.
Die entstandene Datei, welche die Vektoren enthält, wird anschließend verwendet, um ein Modell zu berechnen, das unbekannte HTML-Dokumente (in Form eines Vektors mit der gleichen Struktur) einem der Genres zuordnet. (Abschnitt 3)
In Abbildung 1 wird der Aufbau der Implementierung schematisch dargestellt.

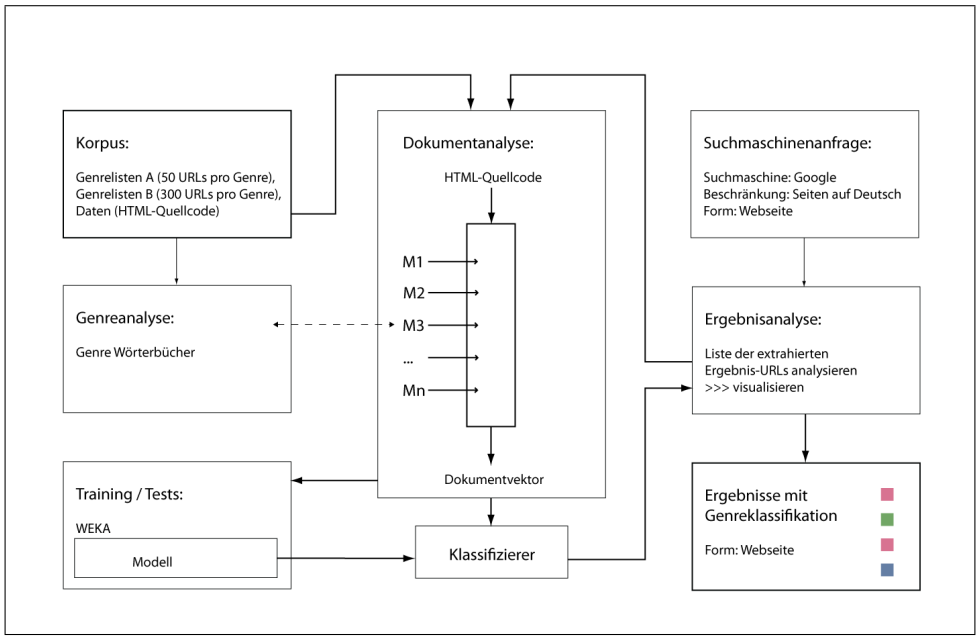

Abbildung 1: Übersicht der Komponenten

[2] Uniform Resource Locator

2.2 Testkorpus

Das Testkorpus besteht aus einer Liste von 300 URLs pro Genre. Dafür wurden eigenhändig eine Liste A aus 50 ausgewählten URLs pro Genre in einer Datei (blogs, ...) gespeichert. Unter Verwendung des *site*[3] Operators von Google werden automatisch für diese 50 URLs jeweils 5 weitere URLs der selben Domäne ermittelt und diese 6 URLs x 50 in einer Datei (blogs_2009-03-13, ...) als Liste B gespeichert.
Abbildung 2 zeigt die ermittelten URLs für die Domäne 'googlewatchblog.de'.

```
http://www.googlewatchblog.de/
http://www.googlewatchblog.de/wiki/
http://www.googlewatchblog.de/forum
http://www.googlewatchblog.de/wishlist
http://www.googlewatchblog.de/archive
http://www.googlewatchblog.de/tag/xoogler/
```

Abbildung 2: Auszug der Datei blogs_2009-03-13

Diese Art der Korpusgenerierung bietet viele Vorteile. Ausgehend von der Hauptdomäne sind die Unterseiten bei weiteren Versuchen nicht relevant, da diese Seiten sich ändern oder nach einer gewissen Zeit nicht mehr existieren. Statische URL-Listen[4] haben das Problem, dass sie so aktuell sind wie 'die Zeitung von gestern'. Durch die hier beschriebene Erzeugung ergibt sich dieses Problem nicht. Die Erweiterung um andere Genres ist sehr einfach, da lediglich eine neue Ausgangsliste erstellt werden muss.

Eine URL wird nur dann in die Liste der zweiten Stufe aufgenommen, wenn sie innerhalb der Implementierung sinnvoll analysiert werden kann. Dazu muss sie folgende Kriterien erfüllen.

1. das Dokument der URL ist vom Typ: text/html (content-type: text/html)
2. die URL gibt keine Fehlermeldung zurück (403 Forbidden, 404 File not found etc.)
3. Soup kann den Quelltext parsen und die Tags *head* und *body* sind nicht leer
4. die Seite verwendet keine Framesets

[3] If you include [site:] in your query, Google will restrict the results to those websites in the given domain. For instance, [help site:www.google.com] will find pages about help within www.google.com. [help site:com] will find pages about help within .com urls. Note there can be no space between the 'site:' and the domain. Quelle: http://www.google.com/help/operators.html

[4] German URL list (49505): `http://corpus.leeds.ac.uk/internet/final-url-de.gz`

Wenn Dokumente Punkt 3 nicht erfüllen, dann können in der Analyse auch keine Merkmale extrahiert werden und somit entsteht eine ungewollte Unschärfe in der Klassifizierung. (siehe Abschnitt **??**)
Am Ende wird für jede URL, aus Liste B, der HTML-Code heruntergeladen und in jeweils einer Datei pro Genre gespeichert. Die Webseiten können offline wesentlich schneller analysiert werden und Modifikationen in der Merkmalsextraktion (Abschnitt 2.3) werden immer auf der gleichen Datenmenge getestet. Einen weiteren Vorteil stellt die Möglichkeit dar, diachrone Analysen durchzuführen.

2.3 Beschreibung der verwendeten Merkmale

In dieser Arbeit findet eine Unterscheidung zweier Arten von Merkmalsgruppen statt: Allgemeine und Spezifische. Die allgemeinen Merkmale sind zum größten Teil Zählungen oder statistische Werte innerhalb eines Dokumentes (Länge des Inhalts, Anzahl der Links etc.) aller Genres. Die spezifischen Merkmale hingegen sind Merkmale, die auf Grundlage von Genrewörterbüchern (siehe unten) für ein Dokument berechnet werden. Auf die Auswahl von Merkmalen, die sich bereits in anderen Arbeiten als ineffizient erwiesen haben (z.B. POS-Tagging in Stein und zu Eissen (2008)), wurde hier verzichtet.

2.3.1 Die allgemeinen Merkmale

Tabelle 1 listet die Originalnamen der Merkmale aus dem Quellcode des Programms auf und an welcher Stelle im HTML-Code sie extrahiert werden.

`html`	htmlStrLength
`head`	headStrLength, headMetas, headLinks, headTitle
`body`	bodyStrLength, links (linkAnchors, linkDomain, linkFulldomain, linkExtern), linebreaks, paragraphs, divs, images, lists (ulists, olists, li), tables (td, tr), formular, input, scripts, styles, comments , headlines, rest, contentStrLength, contentStrLengthSet, averageWordLength, averageWordLengthSet, pointM, commaM, questionM, exclamationM, quotationM, copyrightSign

Tabelle 1: Übersicht der allgemeinen Merkmale

2.3.2 Die spezifischen Merkmale

Bestimmte Worte, Wortgruppen sind für ein Genre dann relevant, wenn sie in den meisten Dokumenten innerhalb dieses Genres auftauchen, aber nicht in den Dokumenten der anderen Genres. Daher wird im Vorfeld, mit Hilfe von Wortfrequenzlisten, ein Wörterbuch für diese Wörter erstellt. Die weitere Verarbeitung gleicht einem 'Spracherkenner', der anhand der Übereinstimmung der Wörter im Wörterbuch mit denen im aktuellen Dokument rät, welchem Genre dieses Dokument zuzuordnen ist.

`head`	Meta (keywords, description)
`body`	Text der Links (a), Text der Listenelemente (li),

Tabelle 2: Übersicht der spezifischen Merkmale

Auszug des Wörterbuches von Meta (keywords, description):
blogs: [(32, 'blog'), (16, 'impressum'), (11, 'wordpress'), (9, 'apple'), (8, 'weblog'), (8, 'google'), (8, 'blogs'), (7, 'iphone'), (7, 'design'), (7, '2008'), (5, 'uns'), (5, 'twitter'), ...],
foren: [(40, 'forum'), (12, 'foren'), (11, 'community'), (9, 'index'), (9, 'hilfe'), (5, 'thema'), (5, 'php'), (5, 'info'), (5, 'fragen'), (5, 'bull'), (5, 'anzeigen'), ...],
nachr: [(29, 'nachrichten'), (18, 'wirtschaft'), (17, 'politik'), (13, 'kultur'), (9, 'zeitung'), (8, 'gesundheit'), (7, 'unterhaltung') ...],
shopp: [(21, 'shop'), (14, 'günstig'), (13, 'kaufen'), (12, 'bestellen'), (11, 'einkaufen'), (10, 'vergleichen'), (10, 'preise'), (10, 'finden'), (9, 'shopping'), (9, 'auswahl'), ...]

Die Klassifizierung unter Verwendung dieser spezifischen Merkmale zeigt sehr gute Resultate (Abschnitt 3.2). Der Unterschied zu Systemen anderer Arbeiten besteht darin, dass diese Listen nicht von Hand erstellt oder manipuliert, sondern automatisch generiert wurden.

3 Die Auswertung mit WEKA

Die Auswertung der Dokumentvektoren erfolgt mit der Waikato Environment for Knowledge Analysis (WEKA), einer Software-Suite der Universität Waikato für maschinelles Lernen, in welcher verschiede Klassifizierungsalgorithmen implementiert sind.

Das Programm teilt die eingegebenen Daten (die Datei mit den Dokumentvektoren) in Trainings- und Testdaten. Die Trainingsdaten werden verwendet, um mit einem bestimmten Algorithmus ein Model zu trainieren und dieses Model die Testdaten klassifiziert.
Die Aufgabe anhand von n-dimensionale Vektoren zu klassifizieren ist stark abhängig von den Daten. Der richtige Algorithmus kann meist nur durch ausprobieren herausgefunden werden (Witten und Frank (2005)). Außerdem ergeben manchmal auch einfachere Algorithmen (z.B. NaiveBayes vs. MultilayerPerceptron) nur gute Ergebnisse, aber dafür in kürzerer Zeit.
Die Daten aus diesem Projekt fallen in den Bereich der Multi-class Dokumentanalyse von n-dimensionalen Vektoren (n ist die Anzahl der Merkmale). Ein sehr gutes und schnelles Verfahren, das einen Support Vektor Klassifizierer trainiert, ist der Sequential Minimal Optimization (SMO) Algorithmus von Platt (1998). Die in WEKA implementierte Version löst das Multi-class Problem, indem es die Klassen paarweise (1-vs-1) vergleicht und auswertet.

3.1 Die Auswertungsergebnisse

Die Daten bestehen aus 1200 Instanzen, je 300 Dokumentvektoren der vier Genres. Die Dokumentvektoren beinhalteten die bereits erwähnten 51 numerischen Werte (Merkmale) und das Genre als nominalen Wert.
Abbildung 3 zeigt die Ausgabe des Ergebnisses in WEKA. Insgesamt wurden mit dem SMO Algorithmus bei 383 von 409 Testdokumenten das richtige Genre identifiziert.

```
=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances         383               93.643  %
Incorrectly Classified Instances        26                6.357  %
Kappa statistic                          0.9152
Mean absolute error                      0.2575
Root mean squared error                  0.323
Relative absolute error                 68.6376 %
Root relative squared error             74.5486 %
Total Number of Instances              409

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.943     0.023      0.935     0.943     0.939      0.97     blogs
                 0.945     0.013      0.963     0.945     0.954      0.984    foren
                 0.866     0          1         0.866     0.928      0.967    nachr
                 0.99      0.048      0.864     0.99      0.922      0.972    shopp
Weighted Avg.    0.936     0.021      0.941     0.936     0.937      0.974

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 100   0   0   6 |   a = blogs
   1 104   0   5 |   b = foren
   5   4  84   4 |   c = nachr
   1   0   0  95 |   d = shopp
```

Abbildung 3: SMO Ergebnisse in WEKA

Die Zeit, um das Model zu berechnen, betrug 0.99 Sekunden und die Identifizierung erfolgte

mit einer minimalen Verzögerung. Im Vergleich dazu: NaiveBayes: 362 (88,5%), 0.06 Sekunden für das Model, Identifizierung sofort vs. MultilayerPerceptron 388 (94,8%), 288,6 Sekunden für das Model, Identifizierung nach ca. 2 Minuten.
Weitere Werte die in der Auswertung dieser Verfahren entstehen sind Precision[5] und Recall[6]. Außerdem wird am Ende eine Konfusionsmatrix erstellt, in der in dieser Analyse kaum, aber bei schlechteren Ergebnissen sehr gut erkennen kann, welches Genre falsch zugeordnet wurde. (shopp falsch identifiziert als blog (6x), als foren (5x), als nachr (4x))

3.2 Die Bewertung der Merkmale

Die allgemeinen Merkmale wurden noch nicht einzeln oder in verschiedenen Kombinationen evaluiert. Jedoch zeigt die Auswertung mit allen allgemeinen Merkmalen 75.0611 % korrekt klassifizierte Instanzen. Dieses Ergebnis zeigt, dass eine Identifikation der Genres aufgrund von statistischer Werte über allen Genreklassen lediglich befriedigende Resultate erzielt.
Unter ausschließlicher Verwendung der spezifischen Merkmale werden 91.4425 % korrekte Klassifizierungen erreicht. (nur Meta: 75.3056 %, nur Links: 77.2616 %, nur Lists: 38.1418 %) Daher ist es sinnvoll auf dieser Grundlage Wörterbücher für weitere Bereiche zu berechnen, wie z.B. des title-Tags im head, den Überschriften (h) oder dem Text eines Dokumentes. Auf diese Art der Analyse des Textes wurde bisher verzichtet, da zusätzliche sprachspezifische Informationen (Stopwortlisten, POS-Tagger, Stemmer, etc.) erforderlich wären, welche das Sytem auf eine bestimmte Sprache beschränken und/oder die Datenmenge insgesamt zu stark zunimmt. Die bereits erwähnte Umfrage ergab, dass für sehr gute Ergebnisse ein erhöhter Zeitaufwand für die Klassifizierung akzeptabel ist, der jedoch 3 Sekunden nicht übersteigen sollte.

4 Fazit

Mit diesem System ist eine gute Grundlage für die Klassifizierung von HTML-Dokumenten geschaffen. Es wurden keine neuen Genres entdeckt oder bestimmt, aber ein sehr flexibles System zur Merkmalsextraktion entwickelt. Die nächsten Schritte wären eine Erweiterung der URL Ausgangslisten und die Ergänzung weiterer Genreklassen. Außerdem müssten die vielen anderen Formate (XML, PDF, SWF ...), die es im Web gibt und hier nicht berücksichtigt wurden, in der Klassifikation enthalten sein.

[5] Verhältnis von richtig erkannten Dokumenten einer Klasse X zur Anzahl aller als X klassifizierten Dokumente
[6] Verhältnis von richtig erkannten Dokumenten einer Klasse X zur Gesamtanzahl der Dokumente von Klasse X

Einige Dokumente sind nicht in den Korpus aufgenommen worden, da ihr Code nicht zu analysieren war. Fehler in HTML-Dokumenten sind keine Seltenheit, wie die Validierung der Quelltexte im Korpus, mit dem WebInterface des W3C[7] Markup Validation Service[8], ergab. Die Ergebnisse sind in Abbildung 4 dargestellt.

- blogs_2009-03-22: 313 Dokumente, fehlerhaft: 278 (Fehler insgesamt: 30249 (maximale Fehleranzahl in einem Doukument: 1341), warnings: 5658), unvalidierbar: 23, perfekt: 12
- foren_2009-03-22: 313 Dokumente, fehlerhaft: 253 (Fehler insgesamt: 22574 (maximale Fehleranzahl in einem Doukument: 1440), warnings: 18856), unvalidierbar: 7, perfekt: 53
- nachrichten_2009-03-22 : 313, fehlerhaft: 273 (Fehler insgesamt: 55680 (maximale Fehleranzahl in einem Doukument: 3487), warnings: 12774), unvalidierbar: 27, perfekt: 13
- shopping_2009-03-22docs: 313, fehlerhaft: 268 (Fehler insgesamt: 50483 (maximale Fehleranzahl in einem Doukument: 2111), warnings: 13049), unvalidierbar: 15, perfekt: 30

Abbildung 4: Ergebnisse der Validierung

Das selbst Dokumente, die nicht in den Korpus aufgenommen wurden, dennoch im Browser angezeigt werden können, liegt an einer sehr robusten Verarbeitung/Interpretation und Reparatur des Quelltextes durch den Browser. Diese Strategien sollten in dem hier vorgestellten System integriert werden, um eine ähnlich robuste Verarbeitung zu erreichen.
Ein anderer Punkt der zu prüfen wäre, ist die Frage, ob Seiten der gleichen Domain zwangsläufig zum gleichen Genre gehören. Unter dieser Annahme konnte zwar eine fast automatische Korpuserstellung ermöglicht werden, die sich stets aus einer aktuellen Linkliste zusammensetzt. Jedoch gab es den Fall, dass eine Nachrichtenseite (Domäne) ein Forum als Unterseite enthielt. Da die Precision für das Genre Nachrichten 1 beträgt, ist sicher, dass diese Seite falsch klassifiziert wurde. Somit ist zu untersuchen, welche Auswirkungen gravierender sind, nicht erreichbare Links statischer URL-Listen oder Querschläger.

4.1 Perspektive - Visualisierung

Abbildung 5 zeigt eine mögliche Visualisierung der Genre-Klassifikation von Suchergebnissen. Die Zuordnung wird durch eine farbige Box vor dem Ergebnis angezeigt. Zusätzlich zum

[7] The World Wide Web Consortium (W3C) is an international consortium where Member organizations, a full-time staff, and the public work together to develop Web standards. `http://www.w3.org/Consortium/`

[8] `http://validator.w3.org/`

Genre (Farbe, Genrebezeichnung) könnten in der Box weitere Informationen stehen, wie z.B. die Genauigkeit mit der der Link zu Genre X gehört. Außerdem besteht die Möglichkeit im Analyseprozesse gewonnene Erkenntnisse auf einer zweiten Ebene zu annotieren, die nur beim 'on-mouse-over' der Box zu sehen sind.
Web-Dokumente, die dieses System nicht analysieren kann (Abschnitt 2.2), würden mit einer grauen Box und dem jeweiligen Grund markiert.

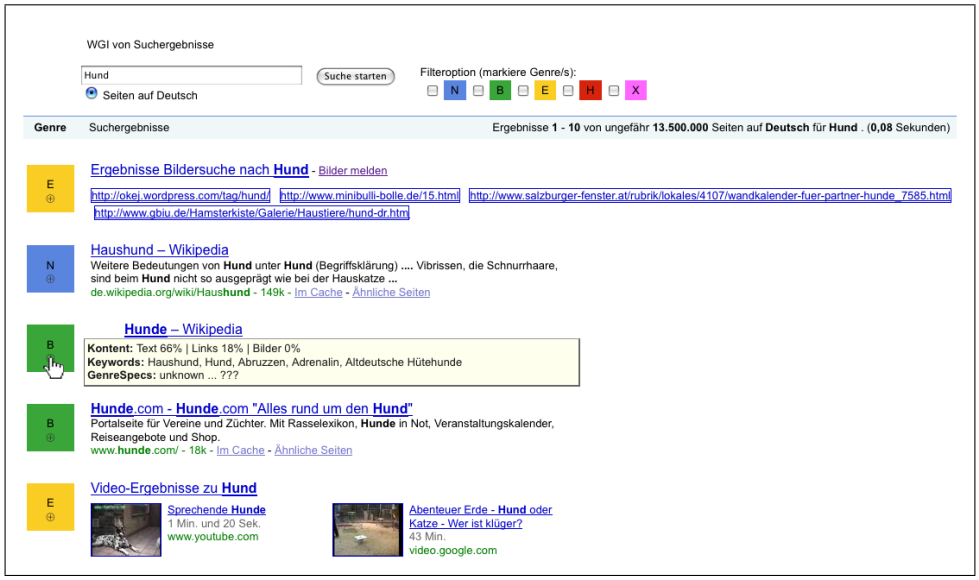

Abbildung 5: Suchergebnisse mit entsprechender Genre-Markierung

Es gibt zwei Möglichkeiten den Suchvorgang unter Verwendung einer Genre-Identifikation/-Klassifikation zu optimieren. Zum Einen kann der Suchende anhand visueller Markierungen die für ihn interessanten Ergebnisse weiterverfolgen, zum Anderen kann die Genre-Identifikation als Filter dienen, so dass nur Ergebnisse präsentiert werden, die zu einem oder mehreren ausgewählten Genre(s) gehören.

Literatur

[Boese und Howe 2005] BOESE, Elizabeth S. ; HOWE, Adele E.: Effects of Web Document Evolution on Genre Classification. In: *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*. New York, NY, USA : ACM, 2005, S. 632–639

[Lahn 2006] LAHN, Andrea: *Genre-Analyse von Web-Dokumenten*, Bauhaus-Universität Weimar, Diplomarbeit, 2006

[Platt 1998] PLATT, J.: *Sequential minimal optimization: A fast algorithm for training support vector machines*. 1998

[Rehm 2006] REHM, Georg: *Hypertextsorten : Definition, Struktur, Klassifikation*, Justus-Liebig-Universität Giessen, Dissertation, 2006

[Soup] SOUP, Beautiful: *Beautiful Soup*. – URL `http://www.crummy.com/software/BeautifulSoup/`. – Zugriffsdatum: Version 3.0.7a, released July 3, 2008

[Stein und zu Eissen 2008] STEIN, Benno ; EISSEN, Sven M. zu: *WEGA - Web Genre Analysis*. 2008. – URL `http://www.benno-stein.de/`. – Zugriffsdatum: 2008

[Witten und Frank 2005] WITTEN, Ian H. ; FRANK, Eibe: *Data Mining: Practical machine learning tools and techniques*. 2nd Edition. San Francisco : Morgan Kaufmann, 2005