HyGraph – Ein Framework zur Extraktion, Repräsentation und Analyse webbasierter Hypertextstrukturen

Rüdiger Gleim

Universität Bielefeld
Universitätsstraße 25
D-33615 Bielefeld

Ruediger.Gleim@uni-bielefeld.de

Typ des Beitrags/Type of the paper Vortrag/Lecture

HyGraph – Ein Framework zur Extraktion, Repräsentation und Analyse webbasierter Hypertextstrukturen

Rüdiger Gleim

Im Hypertextumfeld existiert eine Vielzahl von Verfahren zur Analyse von Hypertexten und –strukturen. Sollen im Rahmen einer Forschungsarbeit verschiedene Methoden kombiniert werden, entsteht jedoch die Problematik, Korpora zu extrahieren, diese für die anvisierten Systeme aufzubereiten und in die meist proprietären Formate zu konvertieren. Das *HyGraph*-System wurde zur Transformation webbasierter Hypertextstrukturen in eine auf der *Graph eXchange Language* basierenden Graphrepräsentation entwickelt. Die repräsentierten Hypertexte können in verschiedenen Sichten bzgl. ihres Inhalts und Struktur exploriert sowie für die Anbindung an weitere Systeme aufbereitet und exportiert werden. Ziel der Entwicklung von *HyGraph* ist die Minimierung des Overheads, die bei der Arbeit mit Hypertextkorpora entsteht.

The field of hypertext research benefits from a multiplicity of procedures for the analysis of hypertexts and -structures. If different methods are to be combined in the context of a research work, however the problem arises to extract corpora from the web and transform them into the proprietary formats of the intended systems. The *HyGraph* system was designed for the transformation of web-based hypertext structures into a generic graph representation which is based on the *Graph eXchange Language*. The represented hypertexts can be explored in various ways concerning their content and structure. Finally there are functions to set up and export the data into several systems for further processing. The most important goal of the development of *HyGraph* is the minimization of the overhead which emerges along the work with hypertext corpora.

1 Einleitung

Webbasierte Hypertexte sind nicht zuletzt wegen ihrer Vielfältigkeit zu einem interdisziplinären und faszinierenden Forschungsgegenstand geworden. In diesem Kontext wirkt insbesondere auch der stetig wachsende Grad ihrer Verbreitung als ein Katalysator für die Entwicklung neuer Methoden in den Bereichen des Information Retrievals und des maschinellen Lernens. Die Verfügbarkeit sehr umfangreicher Korpora ermöglicht die Entwicklung und

Evaluierung von (z.B.) statistischen Methoden in einem Maß wie dies vor wenigen Jahren noch nicht denkbar gewesen ist. Dadurch ist eine Vielzahl von Verfahren implementiert und zugänglich gemacht worden, die wiederum als Ausgangspunkt für weitere Untersuchungen dienen können. Die Erstellung der dazu benötigten Korpora für spezielle Anwendungen ist an sich bereits ein interessantes Forschungsfeld und verschiedentlich untersucht worden [1, 5]. Bei der Zusammenführung von Eingangsdaten und den für die Weiterverarbeitung geplanten Algorithmen ergibt sich jedoch zugleich die Problemstellung, die unterschiedlichen, meist proprietären Formate miteinander zu verbinden. Es existieren bereits Systeme, wie z.B. *Weka* [4] im Bereich des maschinellen Lernens, welche die Implementierung verschiedener Verfahren in sich vereinen und ihren Zugang dadurch deutlich vereinfachen. Die Untersuchungen von Hypertextkorpora bzw. die dafür eingesetzten Methoden sind jedoch so vielfältig, dass sie letztendlich nie von einem einzigen System gekapselt werden können und damit das Problem der Datenkonversion bestehen bleibt.

Das in Java implementierte *HyGraph*-System wurde mit dem Ziel entwickelt, Hypertextkorpora zu Erstellen, auf der Basis der *Graph eXchange Language* (*GXL*) [12] zu repräsentieren und durch Exportfunktionen für weiterverarbeitende bzw. auswertende Prozesse verfügbar zu machen. Darüber hinaus werden unterschiedliche Sichten auf die Daten ermöglicht, um etwa die intellektuelle Kontrolle der verwendeten Klassifizierungs- und Kategorisierungsverfahren zu unterstützen.

2 Repräsentation

Die für das *HyGraph*-System entwickelte Repräsentation von webbasierten Hypertexten zielt auf die Vereinfachung der Anbindung verschiedener, auf Korpora arbeitender Prozesse. Damit der Ansatz für ein möglichst breites Anwendungsspektrum einen Mehrwert erbringt, werden ausgehend von den extrahierten Webseiten mehrere Abstraktionsebenen beschrieben. Diese bauen konsekutiv aufeinander auf und können die möglichen Repräsentationsanforderungen dadurch differenzierter bedienen, als dies durch eine einzige Sichtweise denkbar wäre.

Webbasierte Hypertexte liegen üblicherweise in *HTML* oder *XHTML* vor. Diesen Repräsentationen kann aufgrund des *Document Object Models* (*DOM*) eine sehr detaillierte Ebene der Strukturierung entnommen werden, auf die alle weiteren Sichten aufbauen können. Bei der Abstraktion von dieser Ebene muss

eine Entscheidung über den gewünschten Fokus getroffen werden. In *HyGraph* liegt dieser auf der Verlinkung der einzelnen Konstituenten. Viele der im Netz verfügbaren Webseiten nutzen die Möglichkeit, auf der Basis von Ankern eine seiteninterne Strukturierung aufzubauen, um die Navigation in längeren Texten oder Tabellen zu vereinfachen. Daher wird im ersten Abstraktionsschritt die durch Anker und inzidente Kanten konstituierte interne Webseitenstruktur repräsentiert. Der nächste logische Schritt bezieht die Verlinkung von Ankerknoten mit anderen Webseitenknoten ein, wodurch auf die durch Hyperlinks konstituierte Websitestruktur abstrahiert wird. Dem folgend müsste konsequenterweise in einem abschließenden Schritt die Verlinkung von Websites als abgeschlossene Einheiten untereinander betrachtet werden. Da *HyGraph* den Fokus auf die Betrachtung von Websites legt wird jedoch hier eine Grenze gezogen. Betrachtet man nun dieses Bild ergeben sich mehrere Modellierungsschichten mit zunehmendem Grad der Abstraktion.

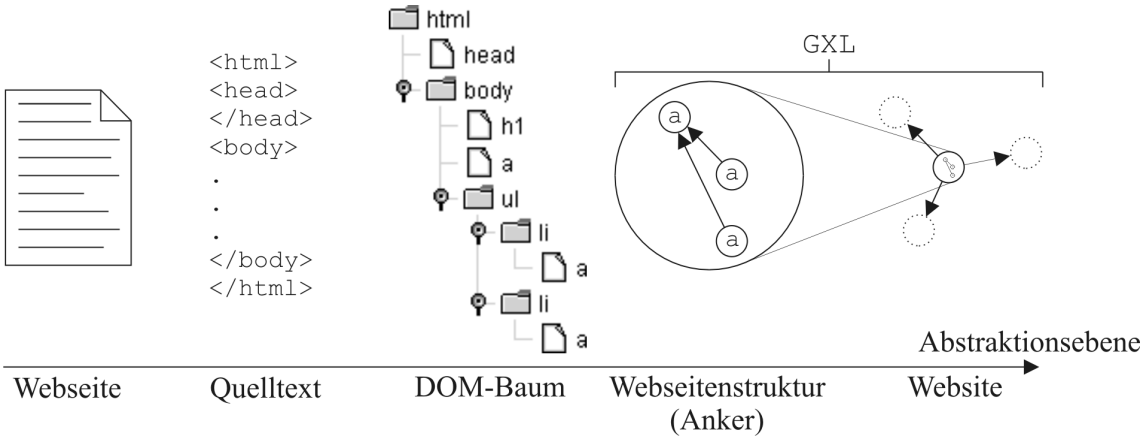

Abbildung 1: Abstraktionsebenen der Repräsentation

Wie kann nun diese hierarchische Abstraktion notiert werden? Die Repräsentation einer Ebene soll möglichst nicht von der darüberliegenden beeinflusst werden. Daher wurde in *HyGraph* auf eine direkte Annotation der markupbasierten Quelltexte verzichtet. Stattdessen wird mit der *Graph eXchange Language* auf eine graphentheoretische Modellierung zurückgegriffen, welche derart komplexe Strukturen darzustellen erlaubt. Die auf *XML* basierende *GXL* wurde als generisches Austauschformat für beliebige Graphen entwickelt. Sie erlaubt insbesondere die Annotation von Graphen, Knoten und Kanten mit Attributen. Darüber hinaus können Knoten selbst wieder Graphen enthalten. Besonders interessant ist schließlich die Möglichkeit, Hyperlinks als Hyperkanten zu repräsentieren. Diese Vorteile sollen anhand der Transformation

einer fiktiven Website in die *GXL* exemplarisch dargestellt werden.

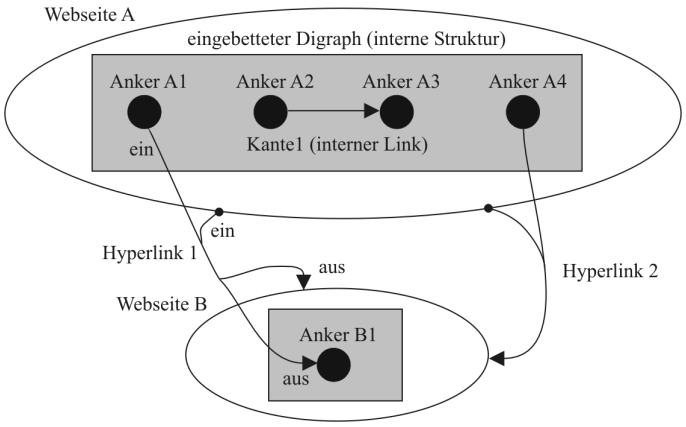

Abbildung 2: Beispiel zur Modellierung einer Website in der GXL

Eine einfache Website bestehe aus zwei Webseiten *A* und *B*. Webseite *A* enthält mehrere Anker (*A1-A4*), die zum Teil untereinander verlinkt sind und damit eine interne Seitenstruktur aufspannen. Die interne Struktur wird als Digraph repräsentiert, welcher die Anker als Knoten sowie die Links als gerichtete Kanten enthält. Im nächsten Abstraktionsschritt liegt der Fokus auf der Verlinkung der Webseiten, die im Beispiel durch zwei Knoten modelliert werden. Diese Webseitenknoten enthalten jeweils einen eingebetteten Graphen, welcher die interne Webseitenstruktur repräsentiert. Die Tatsache, dass im vorliegenden Beispiel zwei Links von der Webseite *A* auf *B* verweisen, könnte nun durch zwei gerichtete Kanten ausgedrückt werden. Dabei geht jedoch die Information verloren, von welchem Anker der Link ausgeht und auf welchen er ggf. verweist. Einfache Kanten im graphentheoretischen Sinne erlauben nur die Verknüpfung zweier Knoten. An diesem Punkt gewinnt die Eigenschaft der *GXL* an Bedeutung, Hyperkanten modellieren zu können, die in der Terminologie der Sprache als Relation bezeichnet werden. Eine Hyperkante kann beliebig viele Knoten miteinander in Beziehung setzen. In der *GXL* wird für jeden Endpunkt notiert, ob er *in die Relation* hinein oder *aus ihr heraus* verweist. Dadurch bleibt die Möglichkeit gewahrt, gerichtete Links darzustellen. In Bezug auf das Beispiel kann nun *Hyperlink 1* derart als eine Relation notiert werden, dass Anker *A1* und Webseite *A* (in die Relation hinein) mit Anker *B1* und Webseite *B* (aus der Relation heraus) in Bezug stehen. Schließlich bietet die *GXL* noch die Möglichkeit, den Endpunkten bestimmte Rollen zuzuweisen. Zur Repräsentation von Hyperlinks sind dies die Start- und Zielwebseite sowie der Start- und Zielanker. Im Beispiel wird bereits anhand von *Hyperlink 2* deutlich, dass der Zielanker optional ist. Das nachfolgende Quelltextfragment fasst die Umsetzung des Beispiels in der *GXL* vereinfacht zusammen. Es soll zudem noch

einmal deutlich gemacht werden, dass die interne Webseitenstruktur in den sie repräsentierenden Webseitenknoten gekapselt ist. Seitenübergreifende Hyperlinks, welche die Websitestruktur konstituieren, werden dagegen auf der nächst höheren Ebene zusammen mit den Webseitenknoten repräsentiert.

```
<gxl xmlns:xlink="http://www.w3.org/1999/xlink">
 <graph hypergraph="true" edgemode="directed" id="Graph0">
  <node id="WebseiteA">
   <graph edgemode="directed" hypergraph="false" id="GraphA">
    <node id="AnkerA1">…</node>
    <node id="AnkerA2">…</node>
    <node id="AnkerA3">…</node>
    <node id="AnkerA4">…</node>
    <edge id="Kante1" from="AnkerA2" to "AnkerA3"/>
   </graph>
  </node>
  <node id="WebseiteB">
   <graph edgemode="directed" hypergraph="false" id="GraphB">
    <node id="AnkerB1">…</node>
   </graph>
  </node>
  <rel id="Hyperlink1">
   <relend direction="in" target="WebseiteA" role="sourcepage"/>
   <relend direction="in" target="AnkerA1" role="sourceanchor"/>
   <relend direction="out" target="WebseiteB" role="targetpage"/>
   <relend direction="out" target="AnkerB1" role="targetanchor"/>
  </rel>
  <rel id="Hyperlink2">…</rel>
 </graph>
</gxl>
```

Im Beispiel wurden Eigenschaften wie etwa *URL*s oder Metatags zur besseren Übersicht ausgeblendet, sie werden jedoch als Attribute in der *GXL* notiert. Für umfangreichere Daten, also insbesondere die *HTML*-Quelltexte der Webseiten, hat sich in der Praxis eine externe Repräsentation als effizienter erwiesen. Zudem hält es die Modellierung in *GXL* schlank und wahrt den Fokus auf hierarchische Strukturen. In *HyGraph* wurde dies derart umgesetzt, dass die *GXL*-Repräsentation zusammen mit weiteren Ressourcen wie z.B. den *HTML*-Quelltexten in einer *Zip*-Datei gekapselt wird. Die Knoten werden mit Verweisen auf die ausgelagerten Ressourcen annotiert. Diese konkrete Implementierung wird durch eine Java-Klasse in *HyGraph* gekapselt und ist für Anwender transparent. Eine Datenbankanbindung wäre somit als Alternative denkbar. Die Einbindung der *GXL* in *HyGraph* erfolgt über die Java Bibliothek *GXL Java-API* [8].

3 Transformation

HyGraph bietet die Funktionalität zur Extraktion von Websites aus dem *WWW* und deren Transformation in die *GXL*-Repräsentation. Neben Einzelextraktionen wird auch die automatische Erstellung von Korpora auf Grundlage einer Liste von Start-*URL*s unterstützt. Die Eingrenzung der Extraktion von Webseiten auf eine Website erfolgt auf der Basis wählbarer Heuristiken. Ausgehend von einer spezifizierten Webseite werden alles Links verfolgt und die referenzierten Seiten untersucht. Erfüllen diese ein gewähltes Kriterium, etwa dass sie auf dem gleichen Host oder im gleichen Unterverzeichnis liegen, werden sie ebenfalls extrahiert. Andernfalls werden sie der Website nicht hinzugerechnet. Dieses Verfahren ist allgemein verbreitet und wird oft als *Webcrawling* bezeichnet. In *HyGraph* wird dazu die frei verfügbare Java Bibliothek *HTMLParser* [11] eingesetzt.

HyGraph arbeitet nach dem Prinzip der kaskadierten Transformation. Ausgehend von einem zunächst leeren Graphen werden alle Elemente sukzessive in einen valides *GXL*-Dokument eingefügt. Über die reinen Strukturinformationen hinaus werden die Knoten und Kanten mit verschiedenen Metainformationen der repräsentierten Elemente annotiert. Die *HTML*-Quelltexte der Webseiten werden zusammen mit dem *GXL*-Dokument gespeichert und sind durch eindeutige Marker referenzierbar.

Zum Abschluss einer Extraktion werden die repräsentierten Hyperlinks typisiert. Das dazu verwendete Verfahren ist eine, auf der Breitensuche basierende Heuristik zur Berechnung der Kernelstruktur. Als *Kernel-Link* typisierte Kanten werden für die Website als strukturbildend betrachtet. Sie bilden einen Spannbaum der Website bei dem alle Webseiten über Baumpfade von der Startseite aus durch gerichtete Kanten erreichbar sind. Die Berechnung der übrigen Linktypen orientiert sich an dieser Struktur. Es sind dies: *Up-Links*, *Down-Links* und *Across-Links*. Zusätzlich werden noch *External-Links* unterschieden welche auf Webseiten verweisen, die gerade nicht mehr der fokalen Website zugerechnet werden. Die nachfolgende Grafik illustriert die verschiedenen Typen.

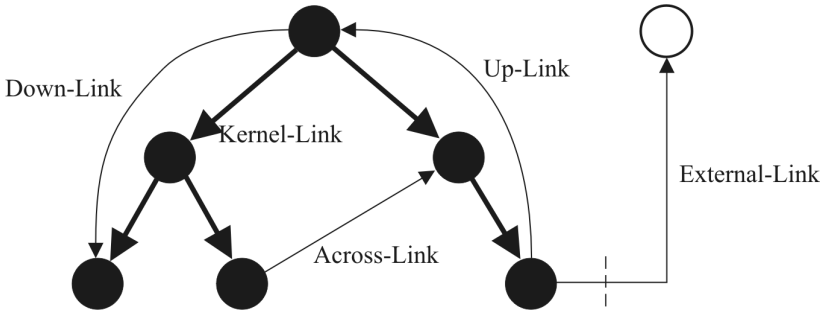

Abbildung 3: Linktypisierung

4 Sichten auf repräsentierte Hypertextstrukturen

Für viele Algorithmen ist eine intellektuelle Kontrolle der berechneten Ergebnisse notwendig um die Performanz der eingesetzten Verfahren und der Parametersätze beurteilen und optimieren zu können (z.B. bei Klassifizierungs- oder Kategorisierungsaufgaben). In *HyGraph* werden verschiedene Sichten auf die repräsentierten Hypertextstrukturen angeboten.

In der „Dokumentansicht“ wird eine Website als ein hierarchisierter Graph zur Navigation der sie konstituierenden Webseiten modelliert. Zu einem gewählten Webseitenknoten werden dann der betreffende *HTML*-Quelltext, die *DOM*-Struktur, die *GXL*-Repräsentation im Quelltext sowie eine *gerenderte* Darstellung angezeigt.

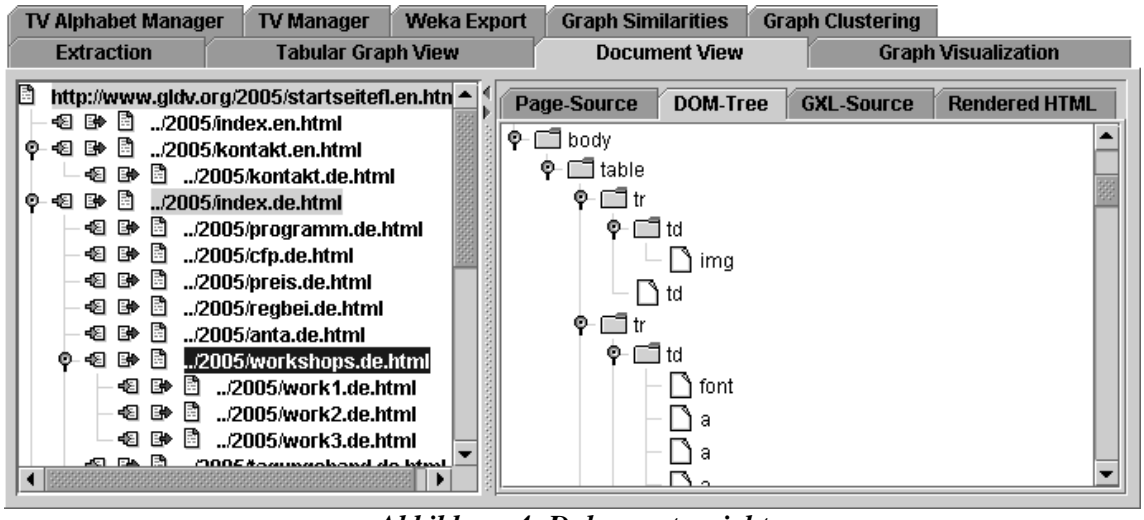

Abbildung 4: Dokumentansicht

Diesen Sichten können alle repräsentierten Informationen entnommen werden. Für große Websites ist es jedoch schwierig sich allein anhand des Navigationsbaums einen Überblick der Sitestruktur zu verschaffen. Daher ist auf der Basis der Open Source Bibliothek *JGraph* [7] eine interaktive Visualisierung implementiert. Die verschiedenen Linktypen sind farblich voneinander abgegrenzt und können wahlweise ausgeblendet werden. Diverse Meta-Informationen der Webseitenknoten sind über Tooltips verfügbar. Schließlich kann über ein Kontextmenü die Dokumentansicht des gewählten Knotens geöffnet werden.

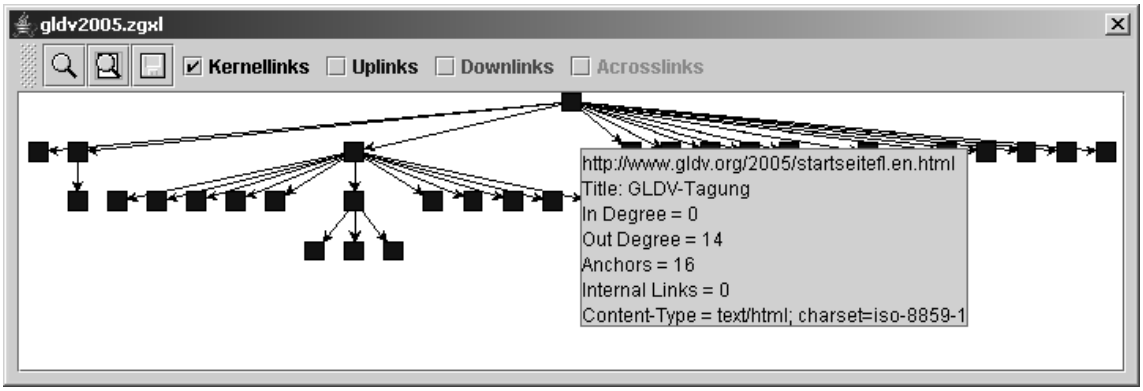

Abbildung 5: Graphvisualisierung

5 Anbindung an Verfahren des maschinellen Lernens

Im Umfeld des maschinellen Lernens existiert eine Reihe von maschinellen Lernverfahren zur Kategorisierung von Texten. Sie basieren zumeist auf der Verteilung der Worthäufigkeiten und müssen zuvor durch überwachtes Lernen trainiert werden. Die Verfahren sind grundsätzlich erfolgreich sowohl für Texte als auch für Hypertextdokumente erprobt worden. Untersuchungen im Hypertextumfeld von Mehler et al. [9, 10] zeigen jedoch, dass klassische Lernverfahren wegen des Phänomen der Mehrfachkategorisierung oft nicht direkt auf Webseiten angewendet werden können.

HyGraph ermöglicht die Berechnung von Worthäufigkeiten und den Export an das *Weka*-System welches verschiedene Verfahren des maschinellen Lernens kapselt. Zusätzlich besteht eine Exportmöglichkeit nach *LibSVM* [6], welches auf dem Prinzip der Support *Vector Machines* arbeitet. Da bei Hypertexten außer Wörtern noch zusätzliche Elementtypen wie z.B. *HTML*-Markup

vorhanden sind und zur Kategorisierung herangezogen werden können, wird im Folgenden der etablierte Begriff des *Token* verwendet.

Zu Beginn der Berechnung der Worthäufigkeiten steht in *HyGraph* die Auswahl der zu berücksichtigenden Tokens. Diese, im Folgenden als *Tokenalphabet* bezeichnete Liste kann aus natürlichen Wörtern aber auch aus *HTML*-Tags bestehen. Die Tokenselektion wird durch eine frei wählbare Klasse durchgeführt, welche die *HyGraph*-spezifische Schnittstelle *ITokenExtractor* implementiert. Auf der Basis eines *HTML*-Dokuments als Eingabe wird die Liste der daraus extrahierten Token als Ausgabe erwartet. Auf diese Weise ist es möglich ohne Neuübersetzung von *HyGraph* weitere Methoden hinzuzufügen. Das Tokenalphabet wird in der Regel auf der Basis derjenigen Websites bzw. Webseiten erstellt, über die später auch die Kategorisierung durch ein maschinelles Lernverfahren erfolgen soll.

Die Auswahl der Token eines Alphabets ist zunächst durch den Tokentyp (Text, *HTML*-Tags etc.) eingeschränkt. Darüber hinaus können Stoppwortlisten sowie Schranken für die *Inverse Document Frequency* (*IDF*) eines Tokens angegeben werden. Dadurch ist es möglich, die im nachfolgenden Schritt berechneten Häufigkeitsverteilungen auf relevante Tokens zu reduzieren, was für das Laufzeitverhalten und die Effizienz von Kategorisierungsverfahren relevant ist.

Nun können die Webseiten ausgewählter Websites mit der Häufigkeitsverteilung bzgl. derjenigen Token annotiert werden, die im dafür angegebenen Tokenalphabet enthalten sind. Da die Verteilungen je nach Umfang des verwendeten Alphabets sehr groß werden können, werden sie nicht direkt im *GXL*-Dokument gespeichert sondern, wie die *HTML*-Quelltexte, als Ressourcen in einer *Zip*-Datei. Die nachfolgende Grafik fasst den Prozess der Tokenalphabeterstellung und der Berechnung der Häufigkeitsverteilung zusammen.

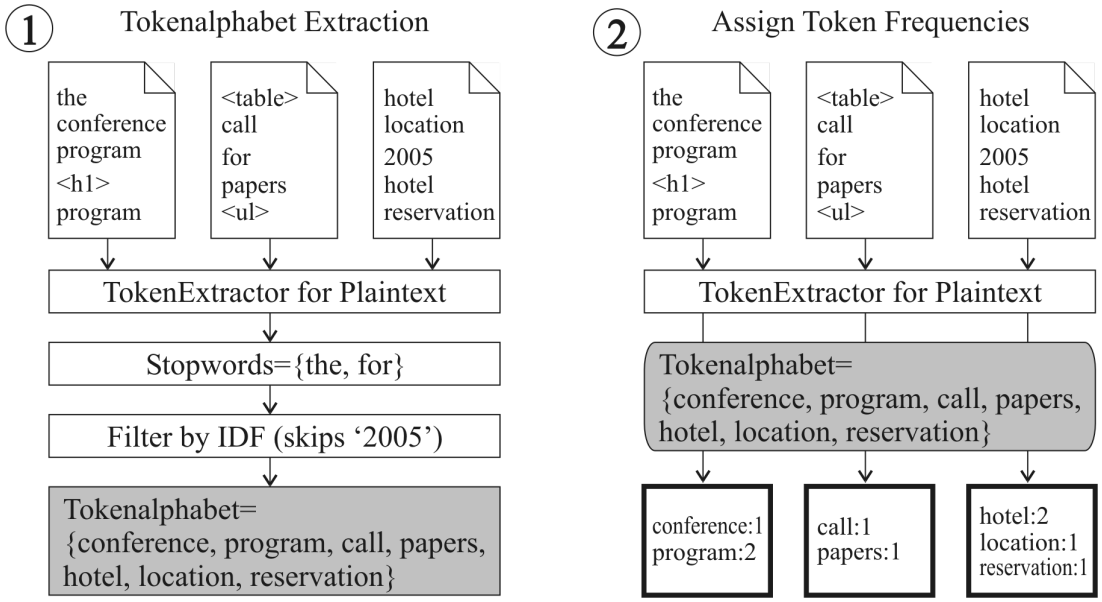

Abbildung 6: Rolle des Tokenalphabets

Auf diesem Weg mit ein oder mehreren Häufigkeitsverteilungen annotierte Webseiten können nun für den Export nach *Weka* oder *LibSVM* ausgewählt werden. Vor dem Export besteht zusätzlich die Möglichkeit, einzelne Webseiten mit Kategorienlabels zu versehen. Diese Funktion kann z.B. für das Training von maschinellen Lernverfahren genutzt werden.

6 Methoden der Strukturanalyse

Das *HyGraph*-System ist in erster Linie zur Erstellung und Exploration von Hypertextkorpora auf der Basis der *GXL* konzipiert, sowie für die Anbindung an weiterverarbeitende Systeme. Mit einem Verfahren von Dehmer et al. [2] ist jedoch auch exemplarisch ein Algorithmus zur Ähnlichkeitsmessung einer bestimmten Graphklasse (nämlich hierarchisierter, knotenmarkierter und gerichteter Graphstrukturen) eingebunden. Das Verfahren wendet *String-Alignments* auf die von den Knoten induzierten Ein- und Ausgangsgradsequenzen der Graphlevel an. Dazu werden zwei Graphen in formale Knotensequenzen, also eindimensionale Strukturen, abgebildet [2]:

$$S_1 = w_1^{H_1} \circ v_{11}^{H_1} \circ v_{12}^{H_1} \circ \cdots \circ v_{1\max_1}^{H_1} \circ v_{21}^{H_1} \circ v_{22}^{H_1} \circ \cdots \circ v_{2\max_2}^{H_1} \circ \cdots \circ v_{h_1 \max_{h_1}}^{H_1}$$
$$S_2 = w_2^{H_2} \circ v_{11}^{H_2} \circ v_{12}^{H_2} \circ \cdots \circ v_{1\max_1}^{H_2} \circ v_{21}^{H_2} \circ v_{22}^{H_2} \circ \cdots \circ v_{2\max_2}^{H_2} \circ \cdots \circ v_{h_2 \max_{h_2}}^{H_2}$$

Die strukturelle Ähnlichkeit zwischen zwei Graphen der oben erwähnten Graphklasse ist nun gleichbedeutend mit einem optimalen Alignment der Sequenzen S_1 und S_2. Spezielle Funktionen zur Bewertung der Alignments detektieren die Ähnlichkeit der Alignments auf den Levels, womit auf die strukturelle Ähnlichkeit zweier hierarchisierter Graphen geschlossen werden kann.

In einem Dialog können die Parameter für den paarweisen Vergleich der gewählten Strukturen eingestellt werden. Dazu zählt insbesondere eine Matrix, welche Distanzangaben zwischen den verschiedenen Knotenmarkierungen erlaubt. Die Art der Eingangs-Struktur ist prinzipiell offen und kann durch die Implementierung eines *HyGraph*-spezifischen Interface erweitert werden. Bereits implementiert ist die Anbindung der durch Webseitenknoten und Kernellinks induzierten Websitestrukturen, sowie den durch das *Document Object Model* gegebenen Webseitenstrukturen. Die so berechneten Ähnlichkeitsgraphen können mit einem ebenfalls exemplarisch implementierten Agglomerativen Clustering ausgewertet und visualisiert werden. Zusätzlich besteht die Möglichkeits des Exports in das Dateiformat des *Markov Cluster Algorithm* (*MCL*) von van Dongen [3].

7 Fazit

Das *HyGraph*-System wurde parallel zu laufenden Forschungsarbeiten entwickelt und verschiedentlich eingesetzt (z.B. [9]). Die Notwendigkeit laufender Erweiterungen machte eine Modularisierung der verschiedenen Funktionseinheiten unabdingbar und hat den Frameworkcharakter des aktuellen Entwicklungsstands geprägt. Mit der *GXL* wird eine universell einsetzbare Sprache zur Darstellung von Graphen für die Repräsentation von Hypertextstrukturen adaptiert. Verschiedene Funktionen wie z.B. die Beschränkung des Extraktionsbereichs und die Ermittlung der Strukturen für den Ähnlichkeitsvergleich sowie die Tokenselektion bei der Berechnung der Häufigkeitsverteilung sind Implementierungen definierter Schnittstellen. Dies ermöglicht die Einbindung weiteren Methoden mit allenfalls minimalen Änderungen am bestehenden Quelltext. Die eingebundene *GXL Java API* und der einheitliche systeminterne Zugriff auf repräsentierte Hypertextstrukturen befreien die Entwicklung neuer Verfahren vom Overhead der Datenakquisition und -haltung.

Der Schwerpunkt bei der Konzeption und Entwicklung lag auf der Repräsentation von Websites als spezielle Ausdrucksform einer Hypertextstruktur. Diese Beschränkung erlaubte eine Optimierung für die Anbindung verschiedener Verfahren des maschinellen Lernens und der Strukturanalyse. Damit ist die vorliegende Arbeit als ein Zwischenschritt hin zu einer universeller einsetzbaren Repräsentation von webbasierten Hypertexten zu verstehen. Durch die praktische Anwendung haben sich sowohl pragmatische, als auch theoretische Aspekte herauskristallisiert, welche in der weiteren Entwicklung berücksichtigt werden. Die Repräsentation soll sich von der Fokussierung auf Websites lösen bzw. allgemeiner verwendbar sein. Desweiteren wird die Integration einer Projektverwaltung notwendig, um die Organisation der Korpora und der darauf arbeitenden Prozesse zu verbessern. Das Ziel ist schließlich eine allgemein zugängliche Plattform für Forschungsarbeiten die auf Hypertextkorpora basieren.

Literaturverzeichnis/References

[1] Baroni M. (2004). BootCaT: Bootstrapping Corpora and Terms from the Web. Proc. of LREC 2004.

[2] Dehmer M., Gleim R., Mehler A. (2004). A new Method of Measuring Similarity for a special class of directed Graphs. Tatra Mountains Mathematical Publications, Slovakia, submitted in August 2004.

[3] van Dongen S. (2000). Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht.

[4] Garner S. R. (1995). WEKA: The waikato environment for knowledge analysis. Proc. of the New Zealand Computer Science Research Students Conference, 57-64.

[5] Ghani R. (2003). Mining the Web to Create Minority Language Corpora. Proc. of 10th CIKM, 279-286.

[6] Hsu C.-W., Chang C.-C., Lin C.-J. (2003). A practical guide to SVM classification. Technical report, Department of Computer Science and Information Technology, National Taiwan University.

[7] JGraph (2005). The JGraph Visualization Library. Web site: http://www.jgraph.com

[8] Larsen E. (2005). Graph eXchange Language Java API. Web site: http://gxl.sourceforge.net

[9] Mehler A., Dehmer M., Gleim R. (2004). Towards Logical Hypertext Structure. A Graph-Theoretic Perspective. To appear in: Proc. of I2CS'04. Berlin-New York, Springer Verlag, 2004.

[10] Mehler et al. (2005). Zur Automatischen Klassifikation von Webgenres. In diesem Band.

[11] Oswald D., Raha S., Kerievsky J., et al.. HTMLParser Java Library. Web site: http://htmlparser.sourceforge.net

[12] Winter, A., Kullbach B., Riedinger V. (2002). An Overview of the GXL Graph eXchange Language. In S. Diehl (Hrsg.), Software Visualization. Berlin: Springer, 324-336.