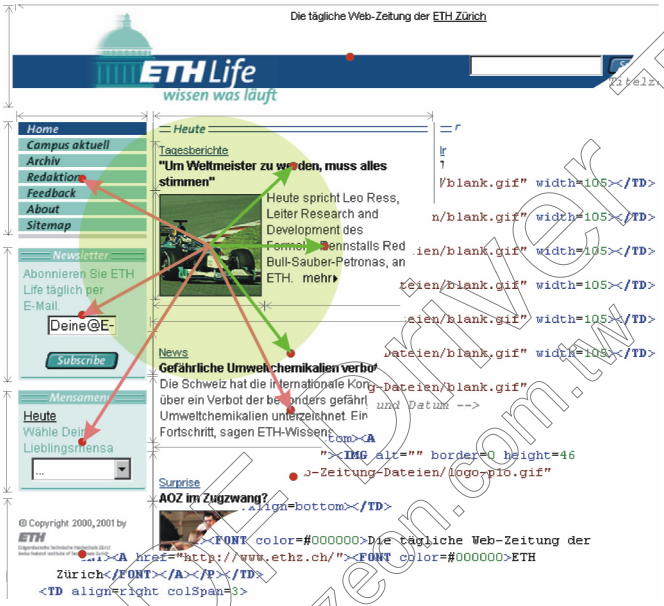
Semesterarbeit

Textextraktion aus HTML-Seiten II



Silvan Saxer, D-INFK

Sommersemester 2001

Institut für Informations-Systeme
ETH Zürich
bei Prof. H.-J. Schek

Betreuer: Michael Mlivoncic und Dr. Roger Weber

1. Abstract

Diese Arbeit führt die Semesterarbeit von Philipp Sieber [Sie01] über Textextraktion aus HTML-Seiten fort. Sie zeigt auf, wie mittels Analyse des grafischen Erscheinungsbilds einer HTML-Seite die darin vorkommenden Bilder automatisch mit geometrisch benachbarten Texten beschrieben werden können. Weiterhin werden die in [Sie01] vorgestellten Merkmale wie Name und Beschreibung des Bildes sowie allgemeine Informationen über das Dokument aus den HTML-Seiten extrahiert.

Neben dem Vorstellen der neuen Komponente wird an Hand von repräsentativen Beispielen untersucht, inwieweit sie der in [Sie01] vorgestellten, einfacheren Variante überlegen ist, und wie gut die Qualität der Extraktion allgemein ist.

Die hier vorgestellte Komponente soll im Rahmen des Projekts „Advanced Querying and Coordination of Multimedia Information in ETH World“ der ETH Zürich eingesetzt werden. Die mit der vorgestellten Methode extrahierten Texte werden dazu an ein Bildsuchsystems weitergeleitet, das diese in Kombination mit grafischen Eigenschaften der entsprechenden Bilder zur Bildsuche verwendet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

2. Einleitung

2.1. Bildsuchsysteme

Zur Suche nach Bildern gibt es grundsätzlich zwei Ansätze.

1. Die Bilder werden mit Begriffen manuell oder automatisch annotiert. Bei der Suche werden die Suchworte mit den gespeicherten Begriffen verglichen und Bilder zurückgegeben, bei denen Suchworte und gespeicherte Begriffe übereinstimmen (automatisch annotiert: [Lyc], von Hand annotiert: [Rin]).

2. Der Inhalt der Bilder wird mittels Fourieranalyse, Farbstatistik, Texturmomenten, etc. analysiert. Als Suchanfrage wird ein Bild angegeben, zu dem das System Bilder findet, deren Inhalt „ähnlich“ ist [Cha]. Solche Systeme werden CBIR (Content-Based Image Retrieval)-Systeme genannt.

Der Hauptnachteil bei der 1. Methode ist, dass die vorgegebenen Stichworte nicht immer mit den späteren Suchbegriffen übereinstimmen. Zudem ist das manuelle Verschlagworten von Bildern äusserst zeitraubend.

Mit Systemen der 2. Art können bereits sehr gute Resultate erzielt werden. Allerdings ist jeweils ein Ausgangsbild notwendig. Oftmals besteht das Problem darin, dass vor der Suche kein geeignetes Ausgangsbild vorhanden ist. In diesem Fall sind solche Suchsysteme unbrauchbar.

Eine vielversprechende Lösung scheint die Kombination der beiden Verfahren zu sein. Allerdings ist noch Gegenstand der Forschung, wie diese beiden unterschiedlichen Merkmale (Textmerkmale, Bildmerkmale) kombiniert werden können. Als mögliche Lösung wird momentan die LSI-Methode vorgeschlagen, die auf der Singulärwertzerlegung von Feature-Image Matrizen beruht ([SLS+01], [DDL+90]).

2.2. „Advanced Querying and Coordination of Multimedia Information in ETH World“

Die Vision von ETH World, dem virtuellen Campus der ETH Zürich (ETHZ), ist es, eine Infrastruktur zu erstellen, die den Zugriff auf die Datenbestände der ETHZ auf einfache Art und Weise ermöglicht. Das Projekt, in dessen Kontext die vorliegende Arbeit anzusiedeln ist, beschäftigt sich mit dem Suchen von Dokumenten und anderen multimedialen Objekten innerhalb der gesamten Dokumentenkollektion der ETHZ.

Leider bestehen momentan an der ETHZ kaum Suchmöglichkeiten für multimediale Daten wie Bilder und Ton- oder Filmdokumente. Gerade Bilder sind jedoch in verschiedensten Fachbereichen (Materialwissenschaften, Mikrobiologie, Forstwissenschaften, ...) besonders relevant.

Durch sinnvolle Kombination und Erweiterung bestehender Systeme soll nun eine Suchinfrastruktur aufgebaut werden, die an bestehende Systeme angebunden wird und mittels eines Webspiders selbständig Daten aus Webseiten extrahiert und verfügbar macht. Dabei ist insbesondere die Erforschung des textuellen Kontexts, in dem die Bilder auftreten, und dessen Relevanz für die Bildsuche von Interesse. Eine detaillierte Beschreibung befindet sich in [IIS01].

2.3. Grafisches Erscheinungsbild von HTML Dokumenten

HTML ist eine Seitenbeschreibungssprache für Internet-Dokumente. Darin werden Informationen über den Inhalt und über die grafische Darstellung (Layout) vermischt codiert. Dabei kann die Positionierung auf dem Bildschirm relativ frei definiert werden. So können beispielsweise Tabellen definiert werden, deren Spalteninhalte logisch zusammengehören, in der HTML-Codierung jedoch an weit auseinanderliegenden Stellen stehen.

Die Befehle für Schriftgrösse, Farbe und spezielle Formatierung wie Fettschrift oder Kursivschrift werden direkt in den Text eingefügt. Diese Formatierung geschieht mittels sogenannten Tags, Kurzbefehlen in spitzen Klammern (Bsp. dieser Text wird fett gedruckt). Zu jedem Tag gibt es ein Endtag (hier), das durch einen Schrägstrich vor dem Kurzbefehl gekennzeichnet ist. Es hebt den vorhergehenden Formatierungsbefehl auf. Offensichtlich entsteht durch diese Codierungsart eine Vermischung von Text und Darstellung.

Der Mensch ist sehr gut im Erkennen von zusammengehörigen Text- und Bildteilen. Allerdings sind die Kriterien, die der Mensch zur Gruppierung verwendet, nicht vollständig erforscht. Insbesondere kann er den Inhalt von Text und Bild verstehen und so inhaltlich unpassende Texte und Bilder einer anderen Gruppe zuordnen. Da zum einen die Grundlagen noch nicht vollständig erforscht sind und zum anderen die Maschine Text und Bildinhalte (noch) nicht verstehen kann, ist eine automatische Gruppierung von logisch zusammengehörenden Text- und Bildteilen auf Heuristiken angewiesen.

Die meisten Ansätze für Bild/Text Extraktion aus HTML Seiten verwenden die Nähe im HTML Code als Gruppierungskriterium [Sieb01,SLS+01]. Infolge der starken Layoutlastigkeit von HTML-Code führt dies jedoch oft zu falschen oder nicht vollständigen Resultaten.

3. Aufgabenstellung

3.1. Aufgabenstellung

Für Bildsuchsysteme ist es sinnvoll, neben den aus dem Bild gewinnbaren Daten wie Farbe und Textur, zusätzlich Stichworte zu verwenden. In einer vorhergehenden Semesterarbeit wurde die automatische Gewinnung von Stichworten aus HTML-Dokumenten betrachtet [Sie01]. Darin wurde gezeigt, dass Texte, die im HTML-Code nahe bei einem Bild vorkommen, oft relevante Information über das Bild enthalten.

In dieser Arbeit wird dieser Gedanke aufgegriffen und verbessert. Da HTML eine sehr Layout bezogene Sprache ist, kommt es manchmal vor, dass Texte, die auf dem Bildschirm nahe beieinander liegen, im HTML-Code einige Entfernung aufweisen. Dies tritt insbesondere bei der Verwendung von Tabellen auf.

Die zu erstellende Komponente soll nun ein auf das tatsächliche Erscheinungsbild (Layout) abgestimmtes Distanzmass verwenden. Damit können den Bildern der HTML-Seite automatisch Texte zugewiesen werden, die auf dem Bildschirm jeweils in der Nähe dargestellt werden.

Nach der Implementierung dieser Komponente soll anhand von Beispielen die Leistungsfähigkeit mit derjenigen der in [Sie01] vorgestellten Methode verglichen werden.

3.2. Ziele

Ziele dieser Arbeit sind:

- Einarbeitung in die in [Sie01] vorgeschlagene Lösung.
- Erarbeiten von Lösungsvorschlägen, wie die Nachbarschaft ermittelt werden kann.
- Entwicklung von Datenstrukturen, mit denen die in der Nähe stehenden Texte einfach gefunden werden können.
- Implementierung einer funktionstüchtigen Komponente zur Textextraktion für das „Advanced Querying and Coordination of Multimedia Information in ETH World“ Projekt.
- Testen der Leistungsfähigkeit im Vergleich mit der in [Sie01] entwickelten Komponente.

3.3. Termine

Folgende Termine sind im Zusammenhang mit dieser Arbeit vorgegeben:

April 2001: Beginn der Arbeiten
Abgabe der Arbeit
Präsentation der Arbeit

4. Extrahierte Informationen

Die Extraktion der Informationen aus dem HTML Dokument geschieht gleichzeitig auf zwei logischen Ebenen, die erst am Ende kombiniert werden: Die Extraktion der Struktur des Dokuments, die für die Distanzberechnung nötig ist, und die Extraktion und Analyse des Textinhalts.

4.1. Strukturextraktion

4.1.1. Aufteilung der HTML Seite in Bereiche

Eine HTML Seite kann man als eine Menge von rechteckigen Bereichen auffassen, die Texte, Bilder oder nichts enthalten können (vgl. Abbildung 1). Die Idee besteht nun darin, die Mittelpunkte der jeweiligen Bereiche zu ermitteln, indem die Grösse der Bereiche abgeschätzt wird. Anhand des Abstands dieser Mittelpunkte kann später die Distanz zwischen einzelnen Bereichen berechnet werden.

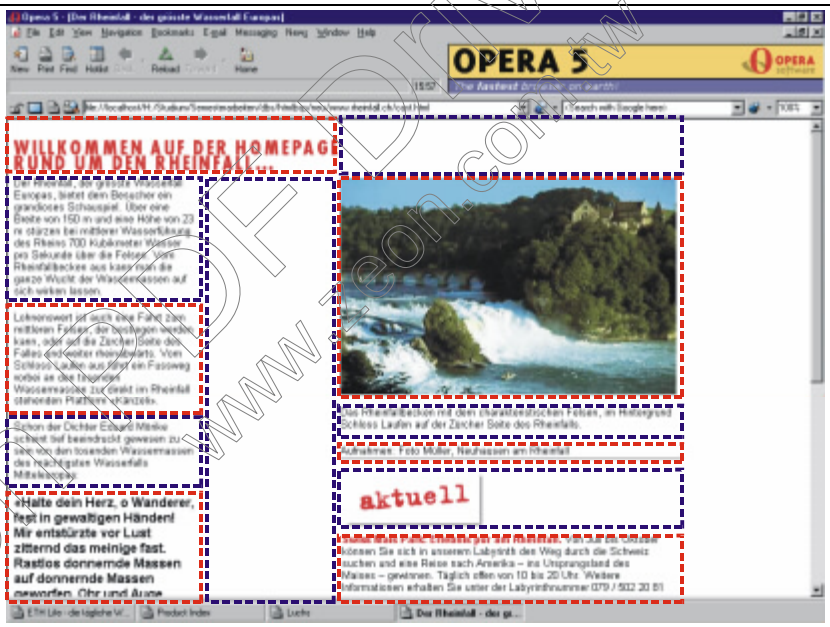


Abbildung 1 : Bereiche einer HTML-Seite, hier als gestrichelte Linien dargestellt

Ein solcher Bereich (im Programmcode AtomicSection genannt) soll so definiert sein, dass er vernünftigerweise nicht weiter unterteilt werden kann. Ein solcher Bereich ist also beispielsweise ein Abschnitt eines Textes oder der Titel eines Kapitels. Insbesondere in HTML definierte Trennungen wie Linien (<hr> Tag in HTML) und Absatzzeichen (<p> Tag in HTML), aber auch lange, dünne Grafiken (Bild einer Trennlinie) trennen Bereiche voneinander.

Diese strenge Definition von Bereichen bedingt, dass eine Zelle einer Tabelle nicht als Bereich gelten kann, da möglicherweise diverse Abschnitte Text enthalten sind (vgl. linke Spalte in Abbildung 1). Es muss daher Bereichsgruppen geben (im Programmcode DividedSection genannt), die frei ineinander schachtelbar sind. Eine Tabelle ist somit eine Bereichsgruppe, die in ihren Feldern wiederum Bereichsgruppen hat (die Zellen der Tabelle). Ebenso ist die HTML-Seite eine Bereichsgruppe.

4.1.2. Mittelpunkt- und Distanzberechnung

Vorraussetzung für die Mittelpunktsberechnung ist das Wissen, wie breit und hoch ein Bereich ist. Leider ist dieses Wissen nicht immer im HTML-Code einprogrammiert. In diesem Fall muss eine Heuristik angewandt werden, da das genaue Berechnen wegen den vielfältigen Gestaltungsmöglichkeiten, die HTML bietet (Schriftgrösse, Schriftart, Schriftdicke, ...), zu aufwändig wäre. In unserem Falle gehen wir von einer (frei definierbaren) Standardschriftgrösse aus. Anhand der Zeichenzahl innerhalb eines Bereichs kann dessen Grösse abgeschätzt werden, sofern eine Seitenlänge des Bereichs (alle Bereiche sind rechteckig) oder das Verhältnis der Seitenlängen gegeben ist. Das Verhältnis der Seitenlängen muss heuristisch bestimmt werden. Es kann beliebig eingestellt werden.

Zur Berechnung der Mittelpunkte werden nun von links oben (0,0) nach und nach alle Bereiche besucht und deren Breite und Höhe addiert, so dass sich die absoluten Werte der links oberen Ecke jedes Bereiches puzzleartig ergeben. Die Berechnung der Mittelpunkte ist anschliessend trivial.

Für jedes Bild, das innerhalb des Dokuments gefunden wurde, werden die euklidischen Distanzen zwischen dem Mittelpunkt des Bereiches, in dem das Bild liegt, und dem Mittelpunkt aller relevanten (vgl. Kap. 4.2.4) Bereiche berechnet. Anhand dieser Distanz werden die am nächsten beim Bild liegenden Bereiche bestimmt. Der in diesen Bereichen enthaltene Text wird dann dem Bild zugeordnet. Die Summe der so erhaltenen Bytes (Buchstaben) kann durch eine obere und eine untere Schranke auf ein sinnvolles Mass festgelegt werden.

Ein weiteres Kriterium, wie viele Bereiche berücksichtigt werden, ist der Wert der Distanz. Er soll sich in einem sinnvollen Bereich befinden. Da die Grösse des Bildes wesentlich zu der Distanz zum nächsten Bereich beiträgt, wurde die maximal zulässige Distanz als n * Bildbreite definiert, wobei n frei wählbar ist. Sobald diese Distanz überschritten wird, wird eine Warnmeldung ausgegeben und nur noch soviel Information gesucht, dass die untere Schranke an Bytes erreicht wird.

4.1.3. Verbesserungsmöglichkeiten

Ein mögliches Problem stellt die Tatsache dar, dass Bereiche durch ihren Mittelpunkt definiert sind. Dies benachteiligt insbesondere grosse Bereiche und Bereiche, die sich über die ganze Seitenbreite erstrecken. Es wäre daher eine (mindestens theoretisch) bessere Lösung, die jeweils kleinste Strecke zwischen zwei Bereichen als Distanzmass zu nehmen.

Abbildung 2 : HTML-Seite mit typischer Musterbildung (Titel, Bild, Kurzmeldung)

Eine weiterer Nachteil ist, dass wiederkehrende Muster im Aufbau der Seite nicht erkannt werden. Abbildung 2 zeigt eine HTML-Seite, die aus diversen Kurzmeldungen mit Titel und Bild bestehen. Wünschenswert wäre es, wenn auch die Maschine diese Gruppierung erkennen würde und damit wüsste, dass der Titel der nächsten Meldung nicht mehr zum Bild gehört. Mit Ansätzen aus der von Max Wertheimer eingeführten Gestalttheorie [Wer24] könnte die Erkennung solcher Gruppen gelingen. Die Gestalttheorie versucht zu beschreiben, wie der Mensch visuelle Merkmale gruppiert. Allerdings gilt zu bedenken, dass die Forschung in diesem Gebiet noch nicht abgeschlossen ist.

4.2. Inhaltsextraktion

Die Inhaltsextraktion stützt sich im Wesentlichen auf die in [Sie01] vorgeschlagenen Merkmale. Es handelt sich daher in diesem Kapitel primär um die Übertragung der Erkenntnisse aus [Sie01] auf Bereiche.

4.2.1. Globale Informationen

Globale Informationen sind Daten, die das ganze Dokument (und damit alle darin enthaltenen Bilder) betreffen. Dieser Extraktionsmechanismus wurde vollständig von [Sie01] übernommen. Er extrahiert die Meta-Tags sowie den Title Tag (Einzelheiten siehe Kap. 4.1. in [Sie01]).

4.2.2. Bildinformationen

Jedes Bild ist in einem Bereich eingebettet. Für die spätere Berechnung der benachbarten Bereiche muss zu jedem Bild abgespeichert werden, in welchem Bereich es sich befindet. Ausserdem wird zu jedem Bild dessen Breite (size) abgespeichert, um die in Kap. 4.1.2. beschriebene maximal zulässige Distanz berechnen zu können.
Wie bereits in [Sie01] werden zudem der Name des Bildes (Dateiname und Pfad) sowie dessen Beschreibung (alt) abgespeichert (vgl. Kap. 4.2.1. in [Sie01]). Da insbesondere wissenschaftliche Texte häufig mit der in HTML vorgesehenen Titelhierarchie (<h1>, <h2>,... Tags) versehen sind, wurde auch dieses Merkmal, das über Bereichsgrenzen hinweg gültig ist, direkt dem entsprechenden Bild zugeordnet (vgl. Kap. 4.2.3. in [Sie01]).

4.2.3. Spezielle Formatierungen

Bereiche bestehen im Wesentlichen aus dem darin enthaltenen Text. Oftmals enthält ein Text jedoch noch zusätzliche Formatierungen wie Fettschrift, Kursivschrift, etc., die einzelne Textbestandteile hervorheben. Da wir annehmen können, dass diese Teile von besonderer Wichtigkeit sind, werden sie separat abgespeichert und können später speziell gewichtet werden (vgl. Kap. 5.5).

Die in dieser Arbeit berücksichtigten Formatierungen sind: Fettdruck (<b>,), Kursivdruck (<i>,) und Unterstrichen (<u>), gleich wie schon in [Sie01] (Kap. 4.2.4. in [Sie01]).

4.2.4. A priori Relevanz eines Textes

Für jeden Bereich wird ein Relevanzwert berechnet. Dieser wird bei der Auswahl der zu betrachtenden Bereiche berücksichtigt. Es kommt oft vor, dass Bereiche entstehen, die keine Information tragen. Insbesondere Menüleisten enthalten meist keine potentiellen Stichworte. Sie lassen sich leicht als eine Häufung von Links erkennen.

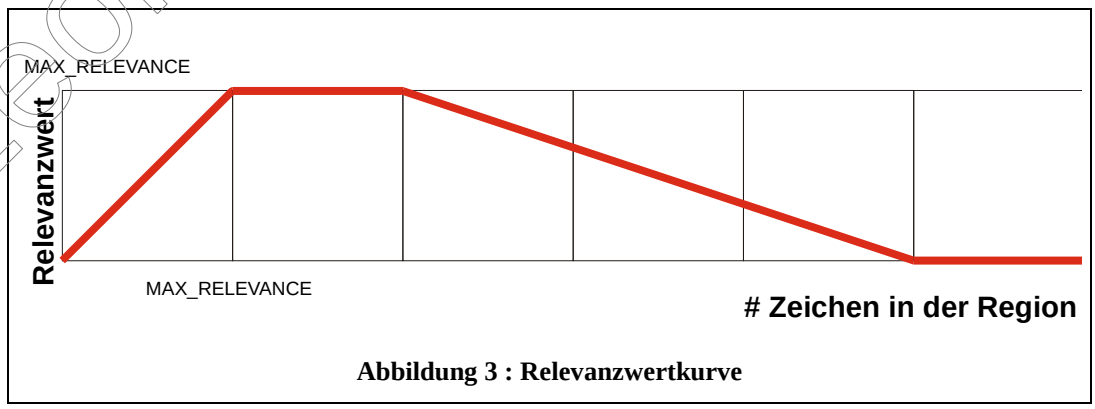


Abbildung 3 : Relevanzwertkurve

Zur Berechnung der a priori Relevanz werden die Länge des Textes sowie die Anzahl Links im Text verwendet. Dabei wird die Kurve in Abbildung 3 verwendet, um eine Relation zwischen Anzahl Zeichen und Relevanzwert zu erhalten. Der maximale Relevanzwert MAX_RELEVANCE kann beliebig gewählt werden. Anschliessend wird der Relevanzwert durch die Anzahl Links im Bereich dividiert, um so die Relevanz von Linklisten (Menüs) zu reduzieren.

5. Implementierung

5.1. Die „Section“ Datenstruktur

Die in Kap. 4.1.1. eingeführten Bereiche und Bereichsgruppen werden in der Datenstruktur „Section“ gespeichert. Sie besteht aus einer abstrakten Oberklasse Section sowie den drei abgeleiteten Klassen AtomicSection, CellSpanSection und DividedSection (vgl. Abbildung 4).

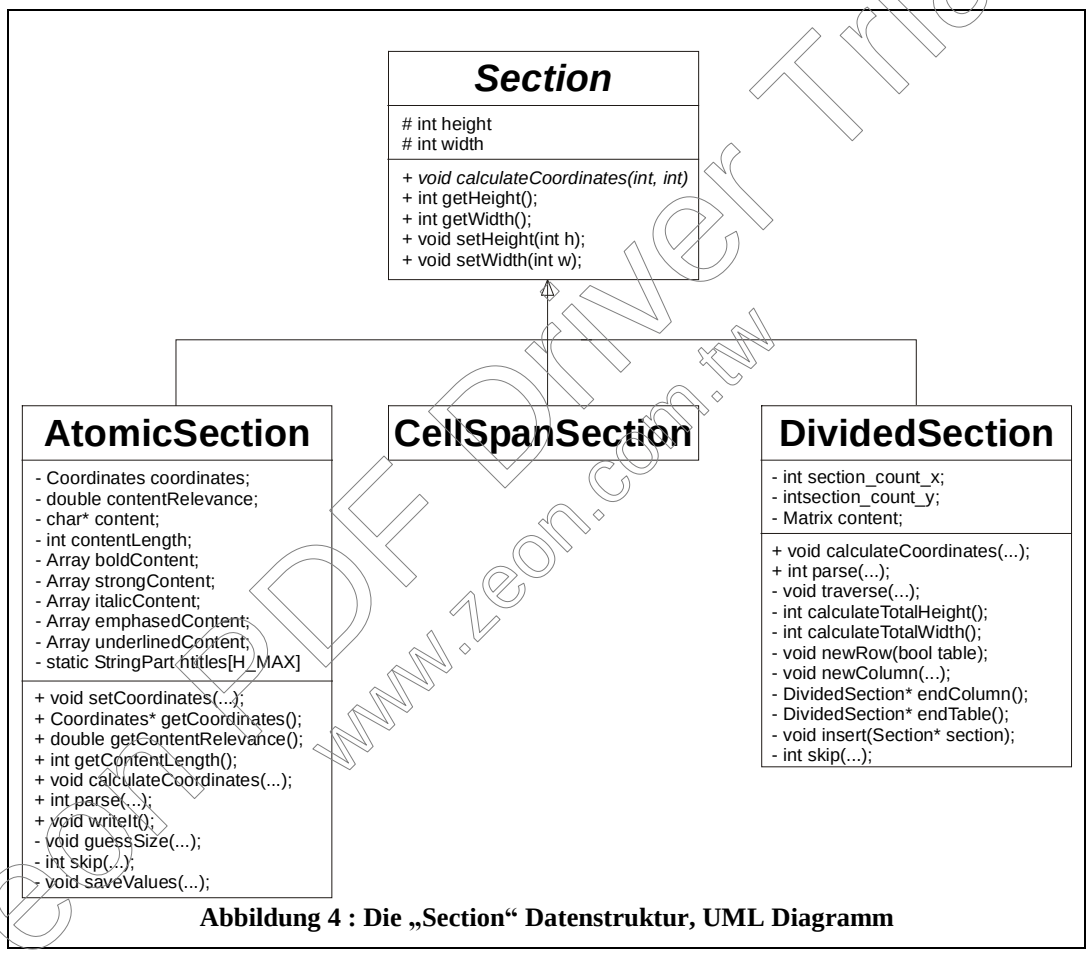


Abbildung 4 : Die „Section“ Datenstruktur, UML Diagramm

5.1.1. Section

In der Section Klasse sind Höhe und Breite des Bereichs gespeichert und protected zugreifbar, d.h. die Subklassen können die Werte lesen und verändern. Zudem ist die Templatemethode calculateCoordinates enthalten, die zur Berechnung der absoluten Koordinate des Mittelpunkts benötigt wird.

5.1.2. AtomicSection

Die Klasse AtomicSection enthält die Daten eines Bereiches. Hier werden die absoluten Koordinaten des Mittelpunkts, die a priori Relevanz, der Inhalt (Text) des Bereichs, die Länge des Inhalts sowie Listen der speziell formatierten Textbestandteile gespeichert. Für einen schön formatierten Ausdruck des Inhalts einer AtomicSection sorgt die Methode writeIt().

5.1.3. CellSpanSection

Die Klasse CellSpanSection ist eine Minimalimplementierung von Section und wird als Platzhalter benötigt, falls im HTML-Code eine rowspan oder colspan Anweisung vorhanden ist (Details: vgl. Kap. 5.2.1.).

5.1.4. DividedSection

Die Klasse DividedSection abstrahiert eine Bereichsgruppe. In ihr sind die Verweise auf die Unterbereiche gespeichert (Variable content). Da wir stets mit rechteckigen, tabellenzellenartigen Unterbereichen arbeiten, kann man die Speicherung in einer Matrix vornehmen.

5.1.5. Beispiel

Abbildung 5 zeigt anhand eines Beispiels, wie eine gegebene HTML-Seitenstruktur in die Datenstruktur „Section“ überführt wird.

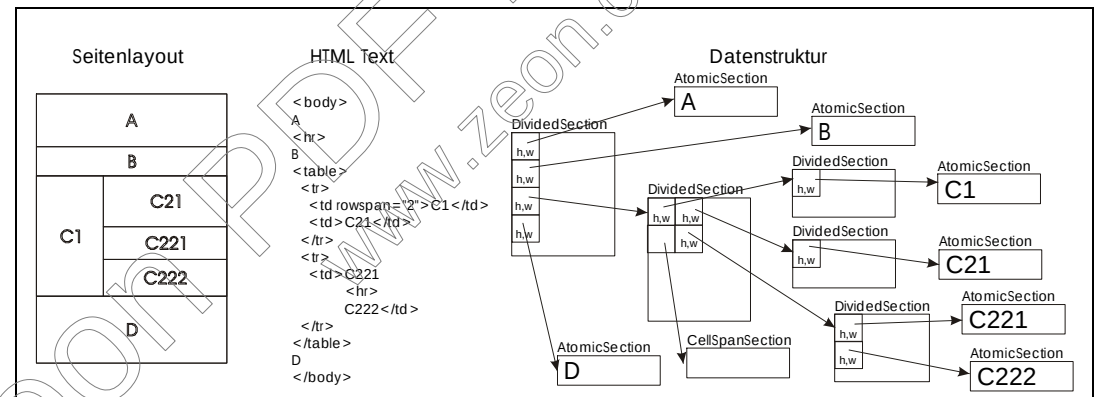


Abbildung 5 : Ein Beispiel für die Datenstruktur „Section“

5.2. Parsen des HTML Dokuments

5.2.1. Aufbau der „Section“ Datenstruktur

Die „Section“ Datenstruktur wird beim Parsen rekursiv aufgebaut. Dies bedeutet, dass jede Klasse der „Section“ Datenstruktur (Ausnahme CellSpanSection) ihren Bereich selbst einliest. Da leider beinahe alle HTML Seiten im Internet (insbesondere solche, die mit automatischen Tools erstellt wurden) nicht in wohlformuliertem XHTML geschrieben sind, gestaltet sich diese Aufgabe äusserst

schwierig. Es kann daher nur ein Teil aller HTML Seiten völlig korrekt eingelesen werden. Es wurde jedoch darauf geachtet, einen möglichst fehlerverzeihenden Parser zu schreiben.

Wie bereits in Kap. 4.1.1. besprochen, wird jede HTML Seite, genauer der <body> der Seite, als DividedSection betrachtet. Demzufolge wird nach dem Parsen des <head> Teils der HTML Seite mittels

```
position = ((DividedSection*)html)->parse(ptr, position, flength,
String("/bo"), false, images, atomicSections);
```

der Parsevorgang für die DividedSection, die die ganze HTML-Seite umfasst, gestartet.

Er funktioniert nach folgendem Prinzip:

- Die aktuelle Position in der Matrix (vgl. Kap. 5.1.4.) wird auf (0,0) gesetzt.
- Die Datei wird zeichenweise gelesen.
- Sobald ein Tag erscheint (Zeichen „<“ wird gelesen), wird dieses ausgewertet:
    - Wird das Endtag der DividedSection (z.B. „/bo“ für </body>) erreicht, so ist der Parsevorgang abgeschlossen.
    - Beginnt eine Tabelle, so wird eine neue DividedSection erstellt und dieser die Kontrolle über das Parsen übergeben (die Tabelle parst sich selbst). Anschliessend wird diese neue DividedSection an der aktuellen Position in die Matrix eingefügt.
    - Erscheint ein <tr> Tag (neue Zeile), so wird die aktuelle y-Position in der Matrix um 1 erhöht.
    - Tritt ein <td> Tag (neue Spalte) auf, wird
        - die aktuelle x-Position der Matrix angepasst.
        - eine neue DividedSection erstellt, die den Inhalt der Spalte parst.
        - diese neue DividedSection wird an der aktuellen Stelle in die Matrix eingefügt.

      Bei Spalten besteht die Möglichkeit, durch Angabe von „rowspan“ und „colspan“ einige Zellen zu verschmelzen. Dies wird hier so gelöst, dass diese Zellen in der Matrix durch Instanzen von CellSpanSection ausgefüllt werden. Damit bleibt die Matrix vollständig gefüllt.
- Andernfalls wird geprüft, ob diese Position allenfalls der Beginn einer AtomicSection sein könnte. Falls ja, so wird
    - eine neue Instanz von AtomicSection erstellt, die sich ebenfalls selbständig parst (vgl. Kap. 5.2.2.).
    - die aktuelle Position in der Matrix angepasst

- geprüft, ob die Relevanz der AtomicSection genügend gross ist, und ggf. wird die AtomicSection in den Array der relevanten AtomicSections eingefügt.
- die AtomicSection in die Matrix an der aktuellen Position eingefügt.

5.2.2. Extraktion des Inhalts einer Atomic Section

Während dem Parsevorgang, werden wichtige Daten abgespeichert. Dazu gehören:

- Beginn und Ende des Textes
- speziell formatierte Textteile (Fett, Kursiv, etc.)
- Titel (H-Tags)
- alle Bilder, die in der AtomicSection vorkommen

Der Parsevorgang funktioniert wie folgt:

- Die Datei wird zeichenweise gelesen.
- Sobald ein Tag erscheint (Zeichen „<“ wird gelesen), wird dieses ausgewertet:
  - Falls es ein Bild-Tag (<img>) ist, wird seine Breite und Höhe ermittelt.
    - dünne, breite Bilder werden als Ende einer AtomicSection interpretiert (grafische Trennlinie)
    - zu kleine Bilder werden aussortiert
    - für jedes „normale“ Bild wird eine Instanz von HTMLImage erstellt (unter anderem werden darin die aktuellen H-Tags gespeichert) und dem Array images hinzugefügt.
  - Falls es ein Tag ist, das das Ende einer AtomicSection bedeutet (<table>, </td>, </body>, <pre>, <code>, , , <hr>, <ol>, <ul>), so werden die Daten der AtomicSection gespeichert (Beginn, Ende, Relevanz, etc.), und der Parsevorgang wird beendet.
  - Falls es ein H-Tag ist, so wird der neue Titel abgespeichert, und alle Untertitel werden gelöscht.
  - Falls es ein Tag ist, das einen Textbestandteil hervorhebt (, , <u>, ,), so wird diese Position vermerkt. Beim erscheinen des Endtags (, , </u>, ,) wird der hervorgehobene Textbestandteil in den entsprechenden Array der aktuellen Instanz von AtomicSection eingefügt.

5.3. Koordinatenberechnung

Nach dem Aufbau der „Section“ Datenstruktur, wird diese rekursiv traversiert, um die Koordinaten jeder AtomicSection zu berechnen. Dabei wird der links obere Punkt der Seite als (0,0) definiert.

```
html->calculateCoordinates(0,0);
```

In der Klasse DividedSection ist die Methode calculateCoordinates wie folgt implementiert:

```
void calculateCoordinates(int xOffset, int yOffset){
 int xoffs;
 int yoffs = 0;

 Section* actSection;
 for (int i = 0; i <= section_count_y; i++){
  xoffs = 0;
  for (int j = 0; content.GetAt(j, i) != NULL; j++){
   actSection = (Section*) content.GetAt(j, i);
   actSection->calculateCoordinates(xOffset+xoffs,yOffset+yoffs);

   xoffs += actSection->getWidth();

  }
  yoffs += actSection->getHeight();
 }
}
```

Dabei werden alle Sections, die in der Matrix content gespeichert sind, rekursiv aufgefordert, ihre Koordinaten zu berechnen. Die Offsets berechnen sich dann jeweils aus der Grösse der Section.

Die Implementierung in AtomicSection berechnet, ausgehend von den angegebenen Offsets, den Mittelpunkt des Bereichs und speichert ihn ab.

```
void calculateCoordinates(int xOffset, int yOffset){
  int xtmp = xOffset+int(this->getWidth()/2);
  int ytmp = yOffset+int(this->getHeight()/2);

  setCoordinates(Coordinates(xtmp, ytmp));
 }
```

Die Implementierung in CellSpanSection ist leer.

5.4. Auswahl der relevanten Bereiche

Für jedes gefundene Bild (gespeichert im Array images) müssen nun die am nächsten liegenden Bereiche gefunden werden. Dazu wird der Abstand zwischen dem Bereich, in dem das Bild liegt, und allen anderen Bereichen (mit genügend

grossem Relevanzwert) verglichen. Dies geschieht durch Einfügen aller Bereiche in eine Prioritätsschlange mit der Distanz als Prioritätswert. Dadurch werden alle Bereiche automatisch nach ihrer Distanz zum Bild sortiert.

```
parentAs = (AtomicSection*) img->getParentSection();
prioq = new PriorityQueue();
for(int j = 0; j < atomicSections->GetNumber(); j++){
  as = (AtomicSection *) atomicSections->GetAt(j);
  dist = (parentAs->getCoordinates()↵
                  ->euklidDist(as->getCoordinates()) / maxdist);
  prioq->Push(as, dist);
}
```

Nach dieser Sortierung geht es darum, ein sinnvolles Mass am Information auszugeben. Dazu wird nach und nach jeweils das oberste Element der Prioritätsschlange ausgelesen und dessen Distanz zum Bild berechnet.

```
bool enoughText = false;
dist = 0;
int nofChars = 0;
while(prioq->MoreData() && !enoughText){
 as = (AtomicSection *) prioq->Pop();
 dist = ((parentAs->getCoordinates()↵
                  ->euklidDist(as->getCoordinates())));
```

Anschliessend wird geprüft, ob bereits die maximale Entfernung vom Bild erreicht wurde (MAX_DISTANCE_FACTOR * Bildbreite). Falls bereits die Minimalanzahl an Bytes gelesen wurde, wird abgebrochen, ansonsten eine Warnmeldung ausgegeben.

```
 if (dist > TablesConstants::MAX_DISTANCE_FACTOR*img->getSize()){
  if (nofChars >= TablesConstants::MIN_CHARS) break;
  LOG("Warning! maximal distance reached");
 }
```

Zuletzt folgt die Ausgabe des Inhalts der AtomicSection mittels der Methode writeIt() (vgl. 5.1.2).

```
 as->writeIt();
 LOG("dist = " + (String) (long) dist);
```

Es folgt die Prüfung, ob die obere Schranke bereits erreicht wurde, anschliessend wird der nächste Bereich aus der Prioritätsschlange geholt (loop).

```
 nofChars += as->getContentLength();
 enoughText = nofChars >= TablesConstants::MAX_CHARS;
} // loop
```

5.5. Ausblick

Die nun erhaltenen Texte pro Bild sollten vor ihrer Verwendung noch weiter verarbeitet werden (vgl. Kap. 5.4. in [Sie01]). Dazu gehört eine Normalisierung, z.B. durch Elimination von Stoppwörtern wie „ist, hat, er, sie,...“ und ggf. einer Stammformreduktion (Stemming). Zudem sollten die Texte in einzelne Wörter aufgespaltet werden. Auch wäre es denkbar, mittels Natural Language Indexing zusammengehörende Terme zu identifizieren, z.B. Substantive und zugehörige Adjekive und Verben. Dies könnte z.B. für ein faktenbasiertes Retrieval nützlich sein ([LST+01]).

Ein weiterer Schritt ist die Gewichtung der einzelnen Wörter (vgl. [SLS+01] sowie Kap. 5.4.2. in [Sie01]). Dabei sind speziell formatierte Textstellen (fett, unterstrichen, kursiv, ...) als besonders relevant zu klassifizieren. Zudem ist es sinnvoll, diejenigen Wörter, die nahe am Bild liegen stärker zu gewichten, als weiter entfernte. Dazu müsste eine sinnvolle Gewichtungsfunktion z.B. der Art

$$w = f(\text{Distanz}) * w'(\text{Auszeichnung})$$

gefunden werden, wobei sich die Wahl von w’() und f() auf den Vorschlag in [SLS+01] stützen könnte.

Anschliessend werden die ermittelten Worte und ihre Gewichte in einer Datenbank gespeichert. Die Gewichte mehrfach vorkommender Worte können addiert werden.

Falls man die Textextraktion verfeinern möchte, wäre es möglich, anschliessend nochmals nach Bereichen zu suchen, die z.B. die drei Wörter mit höchstem Gewicht enthalten. So könnten weitere potentiell relevante Texte extrahiert werden. Möglicherweise könnte man dies gar in einer Art Fixpunktiteration solange wiederholen, bis keine neuen Bereiche mehr gefunden werden und sich so die relevantesten Begriffe herauskristallisiert haben.

6. Resultate

6.1. Allgemeine Bemerkungen

6.1.1. Qualität der Resultate

Die anhand von 14 stichprobenartig ausgesuchten Internetseiten erhaltenen Resultate können überzeugen. Bei beinahe allen geprüften Bildern sind die Texte, die man als Mensch dem Bild zuordnen würde, unter den ersten 3 Resultaten (20 von 22 Bildern (91%)). In 16 Fällen (73%) enthielt bereits der erste Treffer eine passende Beschreibung. Die geometrische Nähe ist offensichtlich ein sehr entscheidendes Merkmal um zu entscheiden, welche Texte für ein Bild relevant sind. Das in [Sie01] als besonders hilfreich bezeichnete Alt-Tag war nur in 7 Fällen (32%) vorhanden und für den Inhalt des Bildes relevant.

Der Titel des Dokuments war ebenfalls nur auf Seiten mit wenigen Bildern hilfreich. Insbesondere bei Internetzeitungen waren sowohl Meta-Tags als auch Titel der Seite für die Bilder nicht relevant. Manchmal können jedoch Titel und Meta-Tags die allgemeine Stossrichung angeben, in welchem Kontext das Bild einzuordnen ist (vgl. Kap. 6.2.5.).

Es hat sich gezeigt, dass die Qualität der Resultate stark von den Werten der Konstanten (einstellbar in der Datei tables.tag) abhängen (vgl. Anhang A). Insbesondere die Parameter `tables.maxchars` (maximale Anzahl Bytes) und `tables.minchars` (minimale Anzahl Bytes) entscheiden wesentlich darüber, ob genügend oder zu viel (falsche) Information gefunden wird. Da dies auch stark vom Informationsgehalt jeder Seite abhängt, muss man einen Mittelweg finden oder aber je nach Typus von zu analysierenden Seiten die Werte separat einstellen.

6.1.2. Problemfälle

Ein Problem stellen nach wie vor (vgl. [Sie01]) grafische Texte dar, oftmals werden gar einzelne Buchstaben als Bild abgespeichert (vgl. Kap. 6.2.3.). Falsch zugewiesener Text tritt meist dort auf, wo der Text zu einem anderen Bild näher ist als der eigene Text, da dieser z.B. grösser ist (vgl. Kap. 6.2.5., Bild „Intimacy“). Ein Ausweg könnten die in Kap. 4.1.3. erwähnten Ansätze „kleinster Abstand statt Abstand der Mittelpunkte“ und „Gestalttheorie“ darstellen. Ebenfalls für unpassende Beschreibungen des Bildes sorgen Texte, zu denen das Bild passt, die aber das Bild nicht beschreiben (vgl. Kap. 6.2.6., Bild „Trommel“, Text über Evelyn Glennie). Manchmal ist der gewünschte Text schlicht nicht vorhanden (vgl. Kap. 6.2.4.: Historisches Museum Basel, gesucht wäre auch: „Barfüsserkirche“).

6.2. Beispiele

Die Beispiele werden einzeln kommentiert. Grau unterlegte Wörter betrachte ich als zum entsprechenden Bild passend.

6.2.1. Beispiel 1

Die 20 Minuten / Auslandseite ist ein typisches Beispiel für die Homepage einer Internetzeitung. Die kleinen Bilder werden jeweils durch einen Titel und einen kurzen Text charakterisiert.

URL: http://www.20min.ch/news/ausland

Globale Information	
Tag	**Text**
Title	20min.ch - Webcenter. Channel News: Nachrichten, Politik, Wirtschaft, Wetter, Ausland, Inland, Bern, Basel, Zürich
Meta.keys	webcenter, 20 minuten, tageszeitung, gratiszeitung, portal, news, nachrichten, neuigkeiten, depechen, depeschen, meldungen, aktualitäten, berichte, schweiz, inland, ausland, zürich, bern, basel, wirtschaft, sport, ski, fussball, eishockey, tennis, digital, vermischtes, kriminell, bilder des tages, wetter, prognosen, dossier, wetter europa
Meta.desc	Das Webcenter der 20 Minuten Tageszeitung. Channel News: Aktuell, Nachrichten, Neuigkeiten. Politik, Wirtschaft, Kriminelles. Schweiz, Regionales und Internationales, Wetter, Sport und Vermischtes. Bilder des Tages. 365 Tage, 18 Stunden, 20 Minuten nach dem Geschehen.

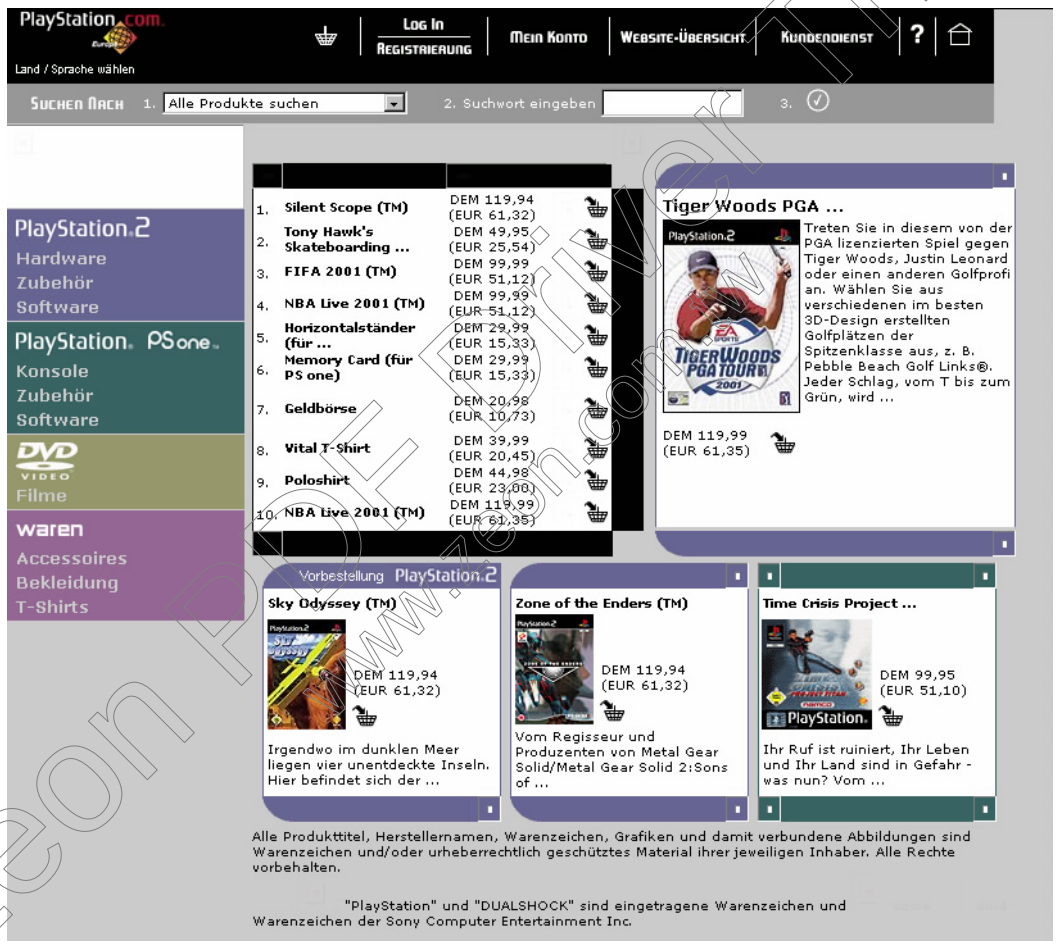
Tag	Distance	Text
Image Name	0	20min-Dateien/bild_tea_dia_shark.jpg
-	156	Haie streicheln ist der neue Touristensport in Australien. Diese Haie bevölkern die Meere. So verhalten Sie sich bei einem Haiangriff.

Tag	Distance	Text
Image Name	0	20min-Dateien/HBKFxBZ2.jpg
- / bold	0	Banditenschicksal
	54	NEU DEHLI - Die frühere indische „Banditenkönigin“ und Parlamentsabgeordnete Phoolan Devi ist ermordet worden. Mehr...

6.2.2. Beispiel 2

Die Playstation-Software Seite zeigt, dass sowohl Text nebenan als auch Text, der unterhalb des Bildes liegt, verwendet wird. Zudem wird mit dem Resultat „Vital T-Shirt“ klar, dass die Abstände nicht immer genau den Abständen in der Wirklichkeit entsprechen. Insbesondere für das „Tiger Woods“ Bild wurde die obere Schranke an Bytes zu hoch gesetzt, da vieles in der Nähe ist, das nicht relevant ist.

URL: http://de.playstation.com/category.jhtml



Globale Information	
Tag	**Text**
Title	PlayStation.com
Meta.keys	navigation, main, products, locate
Meta.desc	Main level of product navigation

Tag	Distance	Text
Image Name	0	P_TIGER_L_EN.jpg
Alt Part	0	Tiger Woods PGA Tour 2001(TM)
	0	Treten Sie in diesem von der PGA lizenzierten Spiel gegen Tiger Woods, Justin Leonard oder einen anderen Golfprofi an. Wählen Sie aus verschiedenen im besten 3D-Design erstellten Golfplätzen der Spitzenklasse aus, z. B. Pebble Beach Golf Links. Jeder Schlag, vom T bis zum Grün, wird ...
- / Bold	134	Tiger Woods PGA ...
- / Bold	136	Vital T-Shirt
	137	DEM 119,99(EUR 61,35)
	161	DEM 39,99(EUR 20,45)
	163	8.
	205	DEM 99,99(EUR 51,12)
	206	DEM 99,99(EUR 51,12)
	212	DEM 29,99(EUR 15,33)
	219	DEM 49,95(EUR 25,54)
	228	DEM 29,99(EUR 15,33)

Tag	Distance	Text
Image Name	0	P_SKYODYSSEY_S_DE.jpg
Alt Part	0	Sky Odyssey (TM)
- / Bold	21	Sky Odyssey (TM)
	94	DEM 119,94 (EUR 61,32)
	117	Irgendwo im dunklen Meer liegen vier unentdeckte Inseln. Hier befindet sich der ...
- / Bold > max dist	208	Zone of the Enders (TM)

6.2.3. Beispiel 3

Die Seite von Sony (Digital Kameras) zeigt ein Beispiel, bei dem wichtige Texte (Typbezeichnungen der Fotoapparate) als grafischer Text geschrieben und daher für das Programm nicht lesbar sind.

Globale Information	
Tag	**Text**
Title	Product Index

Tag	Distance	Text
Image Name	0	dscs50.gif
Alt Part	0	Cyber-shot Digital Still Camera
> max dist	400	Excellent still picture quality with 2.1 megapixels of resolution is only one of this camera's key features. It also records moving images by including MPEG movie mode. ($599 ERSP*)

6.2.4. Beispiel 4

Das Bild der Barfüsserkirche kann nicht vollständig annotiert werden, da nirgends im Text steht, dass das Historische Museum in der Barfüsserkirche untergebracht ist.

URL: http://www.basel.ch/index-ie.php

Globale Information	
Tag	**Text**
Titel	contentpage

Tag	Distance	Text
Image Name	0	101432724442_002.jpg
	270	Der Basler Münsterschatz im Historischen Museum Basel
	274	Vom 13. Juli bis 21. Oktober zeigt das Historische Museum Basel den in der Fälle romanischer und gotischer Goldschmiedewerke einmaligen Kirchenschatz des Basler Münsters.
	314	Bundesfeier am Rhein 2001
	353	Die Openairkino-Saison auf dem Münsterplatz dauert noch bis 11. August 2001.

6.2.5. Beispiel 5

Die Cinema Seite zeigt die Problematik auf, dass der Mittelpunkt eines unpassenden Bereiches manchmal näher beim Bild ist, als derjenige des passenden Bereiches.

URL: http://www.cinema.de/

Globale Information	
Tag	**Text**
Title	Cinema
Meta.keys	cinema, kino, film, movie, kinoprogramm, programm, star, schauspieler, filmmusik, erotik, kult, hollywood
Meta.desc	cinema kino

Tag	Distance	Text
Image Name	0	0,,121887-170-161-ims-ex,00.jpg
	295	Lara Croft - Tomb Raider Erotisch und gefährlich: Angelina Jolie als Lara Croft gibt alles
	317	Angelina Jolie heuerte ihren eigenen Killer an - Hollywoodstar Angelina Jolie (26) hatte eine ... mehr
	372	Quiz Wer raschelt mit dem Rock? Beinharte Fragen, schlaue Antworten im CINEMA-Quiz

Tag	Distance	Text
Image Name	0	0,,108454-76-114-ims-ex,00.jpg
	173	Unverwüstlich Keine alten Kamellen: Wir stellen Rollencharaktere vor, die nie aus der Mode kommen
	220	"Intimacy": Kino der Obsessionen Sex statt Reden, darum geht es in "Intimacy". Ein Blick auf die Höhepunkte am Rande des Abgrunds...

6.2.6. Beispiel 6

Die ETH Life Seite demonstriert den Fall, wo ein Bild zu einem Text passt, der Text jedoch nicht zum Bild (Trommel und Evelyn Glennie). Zudem ist auch bei diesem Bild der Mittelpunkt des beschreibenden Textes weiter weg, als diverse andere Bereiche, was zu falschen Resultaten führt.

URL: http://www.ethlife.ethz.ch/hauptseite

Globale Information	
Tag	**Text**
Titel	ETH Life - die tägliche Web-Zeitung
Meta.keys	ETHLife ETHLive ethlife ethlive webzeitung webmagazine webmagazin ETH Zuerich ETHZ eth ethz
Meta.desc	ETH Life - Die taegliche Web-Zeitung der ETH Zuerich

Tag	Distance	Text
Image Name	0	0,1086,1684,00.jpg
Alt Part	0	bild
	118	Ausserdem heute
	173	Ueber das Bauen mit Computern
> max dist	180	Heute
> max dist	190	Freitag
> max dist	230	Konzert: Evelyn Glennie Montag: Die bekannteste Perkussionistin der Welt. "Musik an der ETH". 19.30 Uhr, ETH Zentrum, HG Semper Aula

6.3. Vergleich zu [Sie01]

6.3.1. Allgemeine Bemerkungen

Die Testkollektion an HTML-Seiten wurde mit beiden Text-Extraktions-Programmen analysiert. Die Resultate sind in Tabelle 1 zusammengefasst.

Tabelle 1 : Vergleich der Text-Extraktions-Algorithmen

Kriterium	WebSpider	TextExtractor
Extrahierter Text enthält mindestens ein treffendes Wort	83 %	100 %
Texte, die man als Mensch dem Bild zuordnen würde, sind unter den ersten 3 Treffern bzw. in AFTER oder BEFORE angeschnitten.	50 %	91 %
Erste Zuordnung ist eine passende Beschreibung bzw. BEFORE oder AFTER enthält eine passende Beschreibung	42 %	73 %
Vollständige Information	8 %	73 %
Keine unpassende Bereiche im Resultat	50 %	41 %
Dokumente erfolgreich geparst	55 %	100 %

Ganz offensichtlich überragt der hier vorgestellte TextExtractor den WebSpider aus [Sie01] in diversen Bereichen (zumindest innerhalb dieser Kollektion). Seine grössten Vorteile sind:

- Es wird stets nach Text gesucht, auch wenn er im HTML Code nicht an einer nahen Byteposition liegt.
- Die gesammelte Datenmenge variiert. Dadurch werden häufiger vollständige Resultate geliefert.
- Es werden jeweils ganze, zusammenhängende Texte (Bereiche) hinzugefügt, was ebenfalls die Qualität der Resultate verbessert.
- Durch die explizite Berücksichtigung von Tabellen, werden auch im Code weiter entfernt liegende Textteile gefunden.
- Das Parsen wurde stark auf Fehlertoleranz ausgerichtet, was zu weniger Abstürzen führt.

Seine grösste Schwäche ist jedoch, dass relativ häufig unpassende Bereiche ins Resultat aufgenommen werden. Dies ist zu einem gewissen Grad durch

Inkaufnahme weniger umfassender Resultate ausgleichbar, ist jedoch zu einem grossen Teil durch das verwendete Prinzip bedingt. Mögliche Abhilfen wurden in Kap. 4.1.3. beschrieben. Allerdings gilt zu bedenken, dass Menschen sehr schnell entscheiden können, ob ein Bild ihrem Bedürfnis entspricht. Daher sind zu viele (und damit möglicherweise falsche) Informationen besser, als wenn ein relevantes Wort nicht berücksichtigt wird. Eine weitere mögliche Schwäche von TextExtractor könnte seine Laufzeit sein. Dies wurde jedoch nicht näher untersucht.

6.3.2. Beispiele

Im folgenden werden die Algorithmen anhand von zwei Beispiele verglichen, um deren jeweilige Vorzüge zu demonstrieren.

6.3.2.1. Beispiel 1

Archos Jukebox 6000
URL: http://www.archos.com/us/products/product_500096.html

TextExtractor findet hier einen andern Text zum Bild. Dieser umschreibt das Produkt besser als der durch WebSpider gefundene Text.

TextExtractor		
Tag	**Distance**	**Text**
Image Name	0	JUKE6000.gif
Alt Part	0	Jukebox 6000 MP3 Player
H4 Part	0	Jukebox 6000 MP3 Player & Hard Drive (PC & Mac)
- / Italic	282	6GB MP3 Player/USB Hard Drive Combo.
	285	
	309	Jukebox 6000 MP3 Player & Hard Drive (PC & Mac)
	339	
	436	Enjoy listening to over 100 hours/6000 minutes of CD-quality music with your ARCHOS Jukebox 6000 MP3 Player/USB Hard Drive. You can conveniently store all your personal files, along with your favorite music selections, and listen to music for 8 hours before recharging the 4 NIMH batteries.

WebSpider	
Tag	**Text**
Image Name	JUKE6000.gif
Alt Part	Jukebox 6000 MP3 Player
Italic	6GB MP3 Player/USB Hard Drive Combo.
Bold	Packages Includes
	Package Includes: Jukebox 6000, MusicMatch Software, USB Interface, A/C Adapter, Stereo Headphones and Pouch. 6GB MP3 Player/USB Hard Drive Combo

6.3.2.2. Beispiel 2

Opel Corsa

In diesem Falls schiesst TextExtractor übers Ziel hinaus und liefert zu viel (unnütze) Information, während die Information von WebSpider genau das Wesentliche umfasst.

TextExtractor		
Tag	**Distance**	**Text**
Image Name	0	a.jpg
	85	Ihr neuer Corsa hat viele Gesichter: Treffen Sie Ihre Auswahl aus vielen attraktiven Lackierungen.
Bold	85	Ihr neuer Corsa
	102	Ihre Auswahl Corsa 3-türig Elegance 1.4 16V ECOTEC Arubablau
Bold	102	Ihre Auswahl
Bold	102	
	105	
	202	Arubablau
	208	Bitte treffen Sie Ihre Auswahl. Klicken Sie dann rechts auf "Weiter".
Bold	208	Bitte treffen Sie Ihre Auswahl.
	249	EURO DM
- / Bold	251	Basispreis
	252	15.635,00 EUR
	261	Basispreis (netto)
Bold	261	Basispreis
	262	Motoren
	264	13.478,45 EUR
- / Bold	272	Mehrpreis
	283	0,00 EUR
	289	Speedster
	292	Ausstattungsvariante

WebSpider	
Tag	**Text**
Image Name	a.jpg
Before	Ihr neuer Corsa hat viele Gesichter: Treffen Sie Ihre Auswahl aus vielen attraktiven Lackierungen.
After	1 : Ihre Auswahl Corsa 3-türig Elegance À 1.4 16V ECOTEC Arubablau

7. Fazit

7.1. Erreichte Ziele

Die Arbeit kann als Erfolg gewertet werden, da alle Ziele erreicht wurden. Es wurde ein Lösungsvorschlag, wie Nachbarschaft beschrieben werden kann, ausgearbeitet, eine entsprechende Datenstruktur entwickelt und alles in einem funktionstüchtigen Programm implementiert. Die erhaltenen Resultate sind vielversprechend und können die Resultate von [Sie01] in mehrfacher Hinsicht verbessern.

Neben den bereits in [Sie01] extrahierten Merkmalen, scheint die geometrische Nähe eines Textes zu einem Bild ein sehr gutes Mittel zu sein, um relevante Texte identifizieren zu können. Trotzdem könnte man wahrscheinlich die Präzision der Antwort noch verbessern, indem man den Inhalt der Seite mit angepassten Methoden aus der Computer Vision analysiert (Gestalttheorie [Wer24]) oder für die Distanzberechnungen nicht nur die Mittelpunkte berücksichtigt.

Unter Umständen könnte durch besseren Miteinbezug des Relevanzwertes (vgl. Kap. 4.2.4.) in die Auswahl der Bereiche die Qualität der Resultate nochmals verbessert werden. Dazu könnte man beispielsweise den Prioritätswert der Prioritätsschlange (vgl. Kap. 5.4.) mit einer Formel wie

$$c_1 * \text{relevance} + c_2 * \text{distance}$$

berechnen, wobei c_1 und c_2 sinnvoll zu bestimmende Konstanten wären.

7.2. Ausblick

Die Extraktions-Komponente, die in dieser Arbeit entwickelt wurde, wird nun in die Spider - Komponente des „Advanced Querying and Coordination of Multimedia Information in ETH World“ Projekts integriert. Schlussendlich sollte ein Web-Crawler entstehen, der selbständig das World Wide Web nach Bildern durchsucht, diese analysiert sowie passende Texte automatisch aus dem umgebenden HTML Dokument extrahiert. Die so gefundenen Daten werden in einer Datenbank abgespeichert und können dann mit einem Suchsystem gefunden werden. Mit dieser Technologie soll ein Bildmanagement innerhalb von ETH World entstehen.

Mit dem erwähnten Web-Crawler könnte die Textextraktion weiter verbessert werden. So wäre es denkbar, die hier für Tabellen vorgestellte Technologie auf Frames auszuweiten. Dazu müssten die Bereiche jeder Frameseite extrahiert werden und anschliessend die Mittelpunkte über alle diese Bereiche berechnet

werden, wobei man die Offsets entsprechend der Position des jeweiligen Frames setzen müsste.

Ein weiterer Erweiterungsschritt wäre die Berücksichtigung der Linkstruktur, wie dies beispielsweise die Suchmaschine Google einsetzt ([BP98]). Allerdings müsste man eine Adaptierung auf Bilder zuerst entwickeln und genauer untersuchen ([AFS01]). Auch wäre es denkbar, den Text, der um einen Link auf das Bild steht, zu verwenden ([AFS01]). Allerdings gilt zu bedenken, dass momentan sehr selten direkte Links auf Bilder zu finden sind, häufiger wird die entsprechende HTML-Seite referenziert.

Anhang A: Empfohlene Einstellung von tables.tag

Alle Beispiele wurden mit der folgenden Einstellung erstellt.

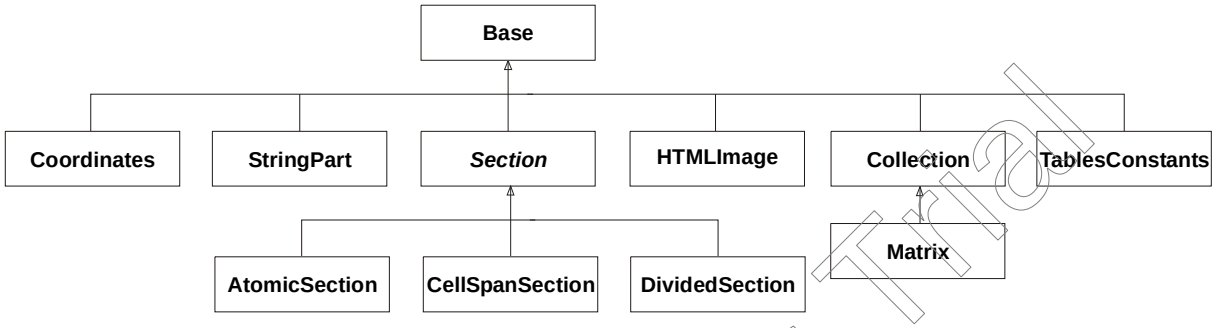
```
tables.additional.distance.table = 0
tables.additional.distance.pre = 2
tables.additional.distance.code = 2
tables.additional.distance.span = 0
tables.additional.distance.div = 0
tables.additional.distance.hr = 200
tables.additional.distance.h1 = 10
tables.additional.distance.h2 = 8
tables.additional.distance.h3 = 6
tables.additional.distance.h4 = 4
tables.additional.distance.h5 = 2
tables.additional.distance.h6 = 0
tables.additional.distance.ol = 0
tables.additional.distance.ul = 0
tables.additional.distance.ul = 0
tables.additional.distance.p = 100
tables.additional.distance.imagebar = 200
tables.image.minsize = 60
tables.image.barwidth = 200
tables.image.defaults.height = 200
tables.image.defaults.width = 200
tables.page.width = 1000
tables.page.height = 600
tables.char.height = 20
tables.char.width = 10
tables.maxrelevance = 1000
tables.sectionheightbywidth = 0.5
tables.contentrelevancethreshold = 0.0
tables.maxchars = 450
tables.minchars = 150
tables.maxdistancefactor = 2.5
```

Anhang B: Referenzen

[AFS01]	V. Athitsos, C. Frankel, M.J. Swani (2001) : „Integrating Analysis of Context and Image Content“, Principles of Visual Information Retrival, Ed. M.S.Lew, Springer
[BP98]	S. Brin and L. Page (1998): „The Anatomy of a Large-Scale Hypertextual web Search Engine“, Proc. 7th Annual World Wide Web Conference (WWW7)
[Cha]	Onlinedemo des ETH Bildsuchsystems CHARIOT http://simulant.ethz.ch/Chariot/
[DDL+90]	S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas and R.A. Harsman (1990): „Indexing by Latent Semantic Analysis“, Journal of the American Society for Information Science 41 p. 391-407
[IIS01]	Institute for Information Systems ETHZ: „Advanced Querying and Coordination of Multimedia Information in ETH Word: Feasibility Study and Prototype Demonstrator“, project description paper http://www-dbs.inf.ethz.ch/download/ETHWorld-Description.ps
[LST+01]	C. Leung, S. So, A. Tam, D. Sutanto, P. Tse (2001): „Semantic-Based Retrival of Visual Data“, Principles of Visual Information Retrival, Ed. M.S.Lew, Springer
[Lyc]	Bildsuchsystem von Lycos http://www.ch.lycos.de/
[Rin]	Bildsuchsystem des Ringier Verlagshauses http://emma.ringier.ch/LightClient/rdb/d_LightClient.htm
[SLS+01]	S. Sclaroff, M. La Cascia, S. Sethi, L. Taycher (2001): „Mix and Match features in the ImageRover Search Enginge“, Principles of Visual Information Retrival, Ed. M.S.Lew, Springer
[Sie01]	Philipp Sieber (2001): „Textextraktion aus HTML-Seiten“, Semesterarbeit am Institut für Informations-Systeme, ETH Zürich
[Wer24]	Max Wertheimer (1924): „Über Gestalttheorie“, Philosophische Zeitschrift für Forschung und Aussprache 1, 39-60 (1925) http://www.geocities.com/HotSprings/8609/gestalttheorie.html

Anhang C: Diagramme

C.1. Klassendiagramm



C.2. Codediagramm