

Bachelorarbeit 2010

Studiengang Wirtschaftsinformatik

Web image crawling and analysis

Crawl | Flickr

Student-in : Jonas Cicognini

Dozent : Henning Müller

Abstract

Deutsch

Diese Bachelorarbeit befasst sich mit der Entwicklung einer Website mit Bildsuche. Das IDIAP Institut benötigt für ihre Entwicklungen, auf dem Gebiet der Bilderkennung, eine umfassende Bilderdatenbank mit den dazugehörigen Notationen.

Im ersten Teil der Arbeit werden die Methoden und Programme vorgestellt, die zu der Realisation der Arbeit benötigt wurden. Folgende Themen werden vorgestellt:

- Die von Flickr zur Verfügung gestellte API zur Entwicklung von Programmen
- PHP Wrapper der auf der API von Flickr aufgebaut ist
- AJAX in Verbindung mit PHP

Der praktische Teil beinhaltet detaillierte Informationen über den Aufbau der Website. Die Website verfügt dabei über folgende Funktionen:

- Suche mittels verschiedener Parameter (Datum, Seitenverhältnis...)
- Erstellung einer Xml Datei der Bilder
- Erstellung einer Zip Datei. Enthalten sind die Bilder sowie die Xml Datei.
- Grafische Vorschau der Suche auf der Website.
- Möglichkeit Bilder aus der Vorschau zu löschen.
- Dynamische Statusanzeige der Suche.

Folgende Kenntnisse sollte der Leser verfügen, um diese Bachelorarbeit zu verstehen:

- Kenntnisse in Html
- Kenntnisse in PHP
- Kenntnisse in JavaScript

Dem Leser soll diese Bachelorarbeit folgendes Know-How vermitteln:

- Der Leser versteht den Aufbau und Funktion der Flickr API.
- Der Leser weiss wie die Flickr API korrekt implementiert wird.
- Der Leser versteht das Konzept von AJAX und ist fähig eigene Aufrufe zu erstellen.
- Der Leser versteht den Aufbau und die Struktur der Onlinesuche "Crawl Flickr"
- Der Leser ist fähig Änderungen an der Onlinesuche "Crawl Flickr" vorzunehmen.

English

Subject of this bachelor thesis is the development of a Website providing an intelligent image search. The IDIAP Institute requires for its developments in the field of image recognition, a comprehensive database of images with the corresponding notations.

In the first part, the necessary methods and programs for the realization of the work are propounded. The following topics are presented:

- The Flickr API
- A PHP Wrapper based on Flickr API
- The use of AJAX with PHP

The practical part includes detailed information on the structure of the site. The Website has the following features:

- Search with different parameters (date,aspect ratio...)
- Creation of an xml file that contains information about images
- Creation of a zip file, containing the xml file and all images
- Graphical preview of the images
- Possibility to delete images from preview
- Dynamic statuspanel, that represents the search progress

The reader should have the following skills to understand this bachelor thesis:

- Knowledge in Html
- Knowledge in PHP
- Knowledge in JavaScript

The reader should earn the following Know how:

- The reader understands the structure and function of the Flickr API.
- The reader knows how the Flickr API is implemented correctly.
- The reader understands the concept of AJAX and is able to create his own calls.
- The reader understands the structure of the online search for "Crawl Flickr"
- The reader is capable to make changes on the online search

Eidesstattliche Erklärung

Ich bestätige hiermit, dass ich die vorliegende Bachelorarbeit alleine und nur mit den angegebenen Hilfsmitteln realisiert habe und dass ich ausschliesslich die erwähnten Quellen benutzt habe. Ohne Einverständnis des Studiengangsleiters und des für die Bachelorarbeit verantwortlichen Dozenten sowie des Forschungspartners, mit dem ich zusammengearbeitet habe, werde ich diesen Bericht an niemanden verteilen.

Brig den 12. August 2010

.....
Cicognini Jonas

Inhalt

Abbildungsverzeichnis	3
1 Einführung	5
1.1 Motivation	5
1.1.1 Motivation des Autor	5
1.1.2 Motivation der Firma	5
1.2 Vorhandene Technologien	5
1.2.1 Google Bildsuche	6
Allgemein	6
Einfache Bildsuche	6
Erweiterte Bildsuche	7
1.2.2 Flickr Bildsuche	8
Allgemein	8
Einfache Bildsuche	8
Erweiterte Bildsuche	9
2 Methoden	11
2.1 Flickr API	11
2.1.1 Flickr API Key	11
2.1.2 Flickr Methoden	12
2.1.3 Methodenaufruf photo.search	13
2.2 PHP Flickr	14
2.2.1 Allgemein	14
2.2.2 Beispiel	14
2.3 Picasa API	14
2.3.1 Allgemein	14
2.3.2 Methodenaufruf	15
2.4 AJAX	15
2.4.1 Allgemein	15
2.4.2 Request Object	15
2.4.3 HTTP Anfrage	16
2.4.4 Handler	17
2.5 zipLib	17

2.5.1	Allgemein	17
2.5.2	Beispiel	18
2.6	Notepad++	18
3	Resultat	19
3.1	Einleitung	19
3.2	Installation	19
3.3	ProgrammoberfläChr	20
3.3.1	Suche	20
3.3.2	Statuspanel	21
3.3.3	Bildvorschau	21
3.4	Programmaufbau	22
3.4.1	Einleitung	22
3.4.2	search.php	22
	Parameter	22
	Funktion	23
3.4.3	analyse.php	23
	Parameter	23
	Funktion	23
3.4.4	getImages.php	24
	Parameter	24
	Funktion	24
3.4.5	createXml.php	25
	Parameter	25
	Funktion	25
3.4.6	zipresult.php	26
	Parameter	26
	Funktion	26
3.4.7	crawlHandler.js	26
	Allgemein	26
	Variablen	26
	Methoden	27
	previewImages	28
	removeImage	29
3.5	Programmablauf	30
3.5.1	Bildvorschau	30
3.5.2	Gesamt	31
3.6	Test	32
3.7	Probleme	32
3.7.1	Anzahl Resultate	32

3.7.2	Löschen temporärer Dateien	32
3.7.3	php.ini	32
3.7.4	Zipdatei	33
4	Schlussfolgerung	35
4.1	Analyse	35
4.1.1	Planung	35
	Soll Planung	36
	Ist Planung	37
	Differenzen	38
4.1.2	Ziele	38
	Image Crawling	38
	Bildernalyse	39
	Webinterface	39
4.2	Zukünftige Arbeit	40
4.2.1	Persönliche Meinung	40
	Literaturverzeichnis	41
A	Annex	43
A.1	Aufgabenstellung	45
A.2	Pflichtenheft	47
A.3	Sitzungsprotokolle	51

Abbildungsverzeichnis

1.1	Google einfache Bildsuche	6
1.2	Google erweiterte Bildsuche	7
1.3	Textuelle Kriterien	7
1.4	Visuelle Kriterien	7
1.5	Flickr einfache Bildsuche	8
1.6	Flickr erweiterte Bildsuche	9
2.1	Flickr API Key Anfrage	12
2.2	Flickr Methoden	12
2.3	Antwort XML File	13
2.4	PHP Flickr Beispiel	14

2.5	Picasa Beispiel	15
2.6	AJAX Request Object	16
2.7	AJAX HTTP Aufruf	16
2.8	AJAX Handler	17
2.9	AJAX Handler	18
3.1	Statuspanel	21
3.2	Bildvorschau	22
3.3	JSON Encode	23
3.4	Url Test	24
3.5	Url Test	25
3.6	Preview Pictures	29
3.7	Preview Pictures	29
3.8	Programmablauf - Preview	30
3.9	Programmablauf - Zipdatei	31
4.1	Soll Planung	36
4.2	Ist Planung	37
4.3	Ziele Image Crawling	38
4.4	Ziele Bilderanalyse	39
4.5	Ziele Webinterface	39

Kapitel 1

Einführung

1.1 Motivation

1.1.1 Motivation des Autor

Für mich stellt das Crawlen von Bildern eine interessante Herausforderung dar. Da ich bis zu diesem Projekt wenig mit dem Thema zu tun hatte und gerne programmiere, wählte ich das Projekt zu meinen Favoriten. Die Tatsache, dass man für einen erfolgreichen Institut (IDIAP) ein Programm entwickelt, war natürlich ein weiterer Anreiz.

1.1.2 Motivation der Firma

Das Projekt war eine Idee von der IDIAP. Diese beschäftigen sich mit der Entwicklung von Algorithmen, die Menschen, Posen und Gesichter erkennen. Damit der Algorithmus effizienter wird, braucht er wie ein Mensch Training. Dafür werden tausende von Bildern benötigt, die korrekt notiert sind. Ein entsprechendes Suchprogramm wurde schon von der IDIAP entwickelt. Dies ist auf der Google Bildsuche gestützt. Durch dieses Projekt möchte die IDIAP ihre Bildersuche erweitern.

1.2 Vorhandene Technologien

Im Internet existieren bereits einige gute Imagecrawler, die öffentlich genutzt werden können. Einige der wichtigsten werden in diesem Kapitel genau erklärt.

1.2.1 Google Bildsuche

Allgemein

Neben einer erfolgreichen Websuche besitzt Google auch über eine Bildsuche. Diese wird regelmässig gewartet und mit den neusten Technologien erweitert. Bei der Google Bildsuche unterscheidet man zwischen der einfachen und erweiterten Bildsuche. Diese Arbeit hätte man auch auf der Google Bildsuche gestützt machen können, da es eine Vielzahl von Bildern bereitstellt und eine gute Suche. Jedoch hat die IDIAP schon ein Projekte das auf der Google Bildsuche basiert.

Einfache Bildsuche



Abbildung 1.1: Google einfache Bildsuche

In der einfachen Bildsuche kann nur ein Tag definiert werden, nachdem gesucht wird.

Erweiterte Bildsuche

Abbildung 1.2: Google erweiterte Bildsuche

In der erweiterten Bildsuche hingegen, gibt es unzählige parameter, die definiert werden können. Hier wird zwischen textuellen und visuellen Kriterien unterschieden.

Abbildung 1.3: Textuelle Kriterien

Textuell Neben einem normalen Tag, kann man hier auch noch definieren was nicht vorkommen darf oder einen genauen Wortlaut, den das Bild haben muss. Durch diese Kriterien kann man seine Suche genauer abgrenzen und erhält ein besseres Resultat.

Abbildung 1.4: Visuelle Kriterien

Visuell Neu hinzugekommen zu der erweiterten Suche ist die Möglichkeit, visuelle Kriterien auszuwählen. Wie im Bild erkennbar, gibt es verschiedene Arten von Bildern, die man auswählen kann. Sei es Gesichter, Fotos, Nachrichten.... Diese Einstufung basiert auf einer Datenbank, welche die entsprechenden Bildern abspeichern und den Inhalt erkennen.

1.2.2 Flickr Bildsuche

Allgemein

Da diese Arbeit auf der Onlineplattform Flickr basiert, wird diese auch erläutert. Bei der Flickr Bildsuche ist es ähnlich wie bei der von Google. Es gibt ebenfalls eine einfache sowie eine erweiterte Suche. Die Suche bietet im allgemeinen aber zu wenig Kriterien um ein gutes Resultat zu erhalten. Dies ist einer der Gründe warum in dieser Arbeit eine eigene Suche entwickelt wird.

Einfache Bildsuche

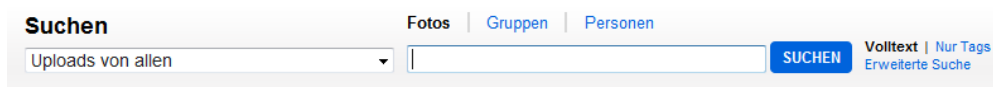


Abbildung 1.5: Flickr einfache Bildsuche

Da Flickr nicht nur eine Bildersuche ist, sondern vielmehr ein soziales Portal, gibt es hier andere Kriterien als in einer reinen Bildsuche. Man kann wählen wo man suchen will was man suchen will (Bilder, User ...) und nach welchem Tag oder Text. Für eine einfache Bildsuche enthält diese schon recht viel und es kann ein genaues Resultat erzielt werden. Dies kann jedoch mit Hilfe der erweiterten Suche noch verbessert werden.

Erweiterte Bildsuche

Erweiterte Suche

Suchen nach

Tipp: Mit diesen Optionen können Sie nach einer genauen Formulierung suchen oder Wörter bzw. Tags von Ihrer Suche ausschließen. Sie können zum Beispiel nach Fotos mit dem Tag "Apfel" jedoch ohne das Tag "Kuchen" suchen.

Alle diese Wörter

☒ Volltext ☐ Nur Tags

Keines dieser Wörter:

Suchen in

Uploads von allen

Sichere Suche

Tipp: Wählen Sie eine "Sicherheitsstufe" für Ihre Suche aus.

☒ Sichere Suche ein ☐ Sichere Suche aus

Nach Inhaltstyp durchsuchen

Tipp: Aktivieren Sie die Kontrollkästchen für die Inhalte, die in der Suche angezeigt werden sollen.

☒ Fotos / Videos
☐ Screenshots / Screencasts
☐ Illustration/Kunst / Zeichentrick/Computeranimation

Nach Medientyp durchsuchen


Tipp: Stellen Sie den entsprechenden Filter ein, um entweder nur Fotos oder Videos in Ihren Suchergebnissen anzuzeigen.

☒ Fotos & Videos
☐ Nur Fotos
☐ Nur Videos
☐ Nur HD-Videos

Nach Datum suchen

Tipp: Verwenden Sie eine oder zwei Datumsangaben, um nach Fotos zu suchen, die in einem bestimmten Zeitraum aufgenommen oder gepostet wurden.

Aufgenommen nach vor
 MM/TT/JJJJ MM/TT/JJJJ MM/TT/JJJJ

 **creative commons**

Tipp: Suchen Sie nach Inhalten mit einer Creative Commons-Lizenz. [Weitere Informationen...](#)

☐ Nur in Inhalten mit einer Creative Commons-Lizenz suchen

☐ Nach Inhalten zur kommerziellen Nutzung suchen

☐ Nach Inhalten für Änderung, Anpassung oder Bearbeitung suchen

Abbildung 1.6: Flickr erweiterte Bildsuche

Die erweiterte Bildsuche enthält neben der einfachen Suche auch noch weitere Kriterien. Neben Datum und Medienart ist die Art der Lizenz eines der wichtigsten Kriterien. Hierbei handelt es sich um die Creative Common Lizenz, es wird unterschieden zwischen Bildern die man für Kommerziellen Projekte nutzen darf oder Bilder die man Nachbearbeiten darf. Mit der Möglichkeit nach solchen Bildern zu suchen, die in dieser Bachelorarbeit gesuchten Bilder werden später in einer Datenbank veröffentlicht, so ist man auf eine solche Lizenz angewiesen.

Kapitel 2

Methoden

In diesem Kapitel werden alle Methoden behandelt, die während der Bachelorarbeit benutzt wurden. Es wird jeweils kurz erklärt wie die jeweilige Methode aufgebaut ist und wie ihre Funktionsweise.

2.1 Flickr API

Die Flickr API wird offiziell von Flickr freigegeben und verwaltet. Sie enthält verschiedene Methoden, die dem Benutzer eine Interaktion mit dem Bildportal Flickr.com erlauben. Auf der offiziellen API Seite von Flickr werden ebenfalls API Kits in verschiedenen Sprachen(PHP,C,Pearl...) angeboten. Diese sind meist von Drittentwicklern und nicht von Flickr selbst. Im Rahmen dieses Projektes wurde ein solches API Kit gebraucht. Dies wird im späteren Kapitel vorgestellt.

2.1.1 Flickr API Key

Damit man eine Methode aufrufen kann, muss man einen API Key besitzen. Diesen kann man als registrierter Benutzer von Flickr in seinem Online Profil anfordern.

Der App Garden

Erstellen Sie eine Anwendung | [API-Dokumentation](#) | [Feeds](#) | [Was steckt hinter dem App Garden?](#)

Zunächst müssen wir wissen, ob Ihre Anwendung kommerziell genutzt wird.

<p>Wählen Sie nicht-kommerziell aus, wenn:</p> <ul style="list-style-type: none"> • Mit Ihrer Anwendung kein Geld verdient wird. • Ihre Anwendung bringt zwar Geld ein, Sie haben jedoch ein familiengeführtes, kleines oder unabhängiges Unternehmen. • Sie entwickeln ein derzeit nicht kommerzielles Produkt, dies könnte sich jedoch in Zukunft ändern. • Sie erstellen eine private Website oder einen Blog, für die Sie nur Ihre eigenen Bilder verwenden. <p>NICHT KOMMERZIELLEN SCHLÜSSEL BEANTRAGEN</p>	oder	<p>Wählen Sie kommerziell aus, wenn:</p> <ul style="list-style-type: none"> • Sie oder Ihr Büro arbeiten für eine bekannte Marke. <p>UND einer der folgenden Punkte ist zutreffend:</p> <ul style="list-style-type: none"> • Sie möchten Gewinne erzielen. • Sie erheben eine Gebühr für Ihr Produkt oder Ihre Dienstleistungen. • Sie beziehen Flickr Inhalte in Ihr Produkt ein und möchten diese Dienstleistungen verkaufen. <p>KOMMERZIELLEN SCHLÜSSEL BEANTRAGEN</p>
--	------	---

Abbildung 2.1: Flickr API Key Anfrage

Wie auf dem Bild ersichtlich, wird zwischen einem kommerziellen und einem nicht kommerziell Key unterschieden. Je nach Projekt wird ein anderer benötigt. In dieser Arbeit wurde ein nicht kommerzieller Key benutzt, da meine Arbeit keinen Profit bringen soll.

2.1.2 Flickr Methoden

Für eine bessere Übersicht wurden die Methoden zusätzlich in verschiedene Gruppen Unterteilt.

photos

- | | |
|--|---|
| <ul style="list-style-type: none"> • flickr.photos.addTags • flickr.photos.delete • flickr.photos.getAllContexts • flickr.photos.getContactsPhotos • flickr.photos.getContactsPublicPhoto • flickr.photos.getContext • flickr.photos.getCounts • flickr.photos.getExif • flickr.photos.getFavorites • flickr.photos.getInfo • flickr.photos.getNotInSet • flickr.photos.getPerms | <ul style="list-style-type: none"> • flickr.photos.getRecent • flickr.photos.getSizes • flickr.photos.getUntagged • flickr.photos.getWithGeoData • flickr.photos.getWithoutGeoData • flickr.photos.recentlyUpdated • flickr.photos.removeTag • flickr.photos.search • flickr.photos.setContentType • flickr.photos.setDates • flickr.photos.setMeta • flickr.photos.setPerms • flickr.photos.setSafetyLevel • flickr.photos.setTags |
|--|---|

Abbildung 2.2: Flickr Methoden

Die wichtigste Gruppe bei dieser Arbeit ist "photos", diese beinhaltet alle Funktionen die man für

eine Bildersuche braucht. Da in dieser Arbeit nach Bildern gesucht wird, ist die Methode "photo.search" die wichtigste.

2.1.3 Methodenaufruf photo.search

Bei dieser Methode gibt es verschiedene Parameter die man definieren kann. Es gibt obligatorische sowie optionale Parameter. Zu den obligatorischen zählt der API Key. Alle anderen sind optional und können frei gewählt werden. Ein paar der wichtigsten optionalen Parameter dieser Arbeit sind:

- **tag:** Ein oder mehrere Wörter nach denen gesucht werden soll
- **tagmode:** Beim tagmode kann zwischen einer AND oder OR-Verknüpfung der Tags gewählt werden. AND bedeutet das ein Bild alle eingegebenen Tags enthalten muss. Bei einer OR-Verknüpfung muss ein Bild jeweils nur einer der gesuchten Tags enthalten.
- **text:** Zusammenhängender Text nachdem gesucht wird. Beim Text wird ebenfalls im Titel und der Beschreibung eines Bildes gesucht.
- **sort:** Es kann nach mehreren Kriterien sortiert werden. z.B. "relevance, date_taken, date_uploaded ..."
- **extras:** Falls in extras nichts angegeben wird, bekommt man eine Standardantwort zurück. Extras erlaubt es Attribute zu definieren, die in der Antwort enthalten sein sollen. z.B.: "description, tags, views ..."

Als Antwort einer solchen Methode erhält man ein XML File. In der Abbildung unten wird ein solche Antwort grafisch abgebildet.

```
<photos page="2" pages="89" perpage="10" total="881">
  <photo id="2636" owner="47058503995@N01"
    secret="a123456" server="2" title="test_04"
    ispublic="1" isfriend="0" isfamily="0" />
  <photo id="2635" owner="47058503995@N01"
    secret="b123456" server="2" title="test_03"
    ispublic="0" isfriend="1" isfamily="1" />
  <photo id="2633" owner="47058503995@N01"
    secret="c123456" server="2" title="test_01"
    ispublic="1" isfriend="0" isfamily="0" />
  <photo id="2610" owner="12037949754@N01"
    secret="d123456" server="2" title="00_tall"
    ispublic="1" isfriend="0" isfamily="0" />
</photos>
```

Abbildung 2.3: Antwort XML File

Wie in der Abbildung ersichtlich ist, werden nur die Standardattribute zurückgeliefert, wenn nichts anderes definiert wird.

2.2 PHP Flickr

2.2.1 Allgemein

PHP Flickr ist ein sogenanntes API Kit von Flickr. Es handelt sich um einen PHP wrapper, der von Dan Coulter entwickelt wurde. Der Wrapper ist komplett in php geschrieben. Die Methoden wurden dabei so entwickelt, dass man jeweils ein php Array mit dem entsprechenden Resultat erhält. Dies ermöglicht ein effizientes und intuitives Arbeiten, selbst ohne Vorkenntnisse im Umgang mit der Flickr API.

2.2.2 Beispiel

```
...  
require_once("phpFlickr.php");  
$f = new phpFlickr("<api key>");  
  
$recent = $f->photos_getRecent();
```

Abbildung 2.4: PHP Flickr Beispiel

Das Beispielskript im obigen Bild zeigt, wie man mit Hilfe der PHP Flickr Klassen eine Abfrage ausführen kann. Im ersten Schritt wird die Datei "phpFlickr.php" eingebunden. Diese Klasse enthält alle Funktionen die man benötigt. Nach dem Einbinden der Klasse wird ein Objekt erstellt. Mithilfe dieses Objekts können nun alle Funktionen der eingebundenen Klasse direkt angesprochen werden. Im letzten Schritt des Skripts wird die Funktion `photos_getRecent` ausgeführt. Anders als bei direkten Aufruf einer Flickr Methode erhält man bei dieser Funktion ein Array als Antwort. Das Parsen der XML geschieht automatisch in der php Funktion.

2.3 Picasa API

2.3.1 Allgemein

Da die Picasa API auch eine mögliche Wahl für diese Arbeit gewesen ist, wird diese hier auch kurz vorgestellt. Picasa ist ein Produkt von Google und stellt eine ähnliche Bildportal wie Flickr dar. Ebenso besitzt Picasa über eine API, die in verschiedenen Programmiersprachen gebraucht werden

kann. Anders als bei der Flickr Authentifikation wird bei Picasa ein Google Account benötigt.

2.3.2 Methodenaufruf

Auf dem nächsten Bild ist ein Aufruf zu sehen, welcher alle Bilder eines Albums holt. Dabei wird eine Variable *\$Query* erstellt, welche alle Parameter der Suche enthält. Dieses wird ausgeführt und anschliessend werden die Bilder mittels einer *foreach* Schleife einzeln ausgegeben.

```
// Creates a Zend Gdata Photos AlbumQuery
$query = $gp->newAlbumQuery();

$query->setUser("default");
$query->setAlbumName("sample.albumname");

$albumFeed = $gp->getAlbumFeed($query);
foreach ($albumFeed as $albumEntry) {
    echo $albumEntry->title->text . "<br />\n";
}
```

Abbildung 2.5: Picasa Beispiel

Da die API von Picasa kein Teil dieser Bachelorarbeit ist, wird diese hier nicht weiter erläutert.

2.4 AJAX

2.4.1 Allgemein

AJAX bedeutet "Asynchronous JavaScript and XML". Es bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Browser und dem Server. Dieses ermöglicht es, HTTP-Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden. Da AJAX in dieser Arbeit häufig gebraucht wurde, wird hier genau erklärt wie ein solcher Aufruf funktioniert.

2.4.2 Request Object

Damit man HTTP Anfragen per AJAX schicken kann braucht man ein sogenanntes Request Object. Die Funktion welche ein solches Objekt erstellt, wird im nächsten Bild gezeigt.

```
/**
 * This function creates a RequestObject depending on the Browser
 * @return RequestObject
 */
function createRequestObject() {
    var ro;
    var browser = navigator.appName;
    if(browser == "Microsoft Internet Explorer"){
        ro = new ActiveXObject("Microsoft.XMLHTTP");
    }else{
        ro = new XMLHttpRequest();
    }
    return ro;
}
```

Abbildung 2.6: AJAX Request Object

Die obige Funktion gibt ein Request Objekt zurück. Da die heutigen Browser nicht alle gleich sind muss für den Internet Explorer ein anderes Objekt erstellt werden. Darum wird in der Funktion unterschieden zwischen Internet Explorer oder anderen Browsern.

2.4.3 HTTP Anfrage

Als nächstes kommt die Funktion, welche unseren HTTP Request sendet.

```
http = createRequestObject();
//define ajax handler
http.onreadystatechange = searchPhotosHandler;
//define ajax-PHP file
http.open('POST', 'search.php', true);
//define all parameters
parameters = 'tag='+tag+'&fulltext='+fulltext+'&tagmode='+tagmode+'&sdate='+sdate;
//HTTP Header
http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
http.setRequestHeader("Content-length", parameters.length);
http.setRequestHeader("Connection", "close");
//send
http.send(parameters);
```

Abbildung 2.7: AJAX HTTP Aufruf

Zuerst wird ein Request Object erstellt, dies geschieht mit der vorherigen Funktion. Danach wird ein Handler bestimmt, welcher die Antwort bearbeitet. Anschliessend definiert man das Skript welches aufgerufen werden soll (search.php). Für die Datenübertragung kann man zwischen der Post oder der Get Methode wählen. Ebenfalls kann man Parameter definieren welche übertragen werden sollen. Als letztes werden die gesammelten Informationen mit dem Befehl http.send abgeschickt.

2.4.4 Handler

Um die Antwort zu verarbeiten wird ein Handler für die Funktion erstellt.

```
function searchPhotosHandler()
{
    var statusdiv = document.getElementById('statusText');
    if(http.readyState == 4){
        var rT = http.responseText;
        //saving image informations
        photoInformations = eval('(' + rT + ')');
        statusdiv.innerHTML = statusdiv.innerHTML + '\n'+photoInformations.length + " hit(s)";
        if(photoInformations.length != 0)
        {
            startanalysis();
        }
        else
        {
            disableLoadGif();
            statusdiv.innerHTML = 'No results';
        }
    }
    else
    {
        statusdiv.innerHTML = 'Searching...';
    }
}
```

Abbildung 2.8: AJAX Handler

In diesem Handler kann man je nach readyState die Daten bearbeiten. Dabei kann der readyState mehrere Werte annehmen. Der wichtigste ist 4, dies bedeutet das die Anfrage komplett ist. In der Variabel Responsetext ist die Antwort des Skripts gespeichert. Dieses kann dazu gebraucht werden Resultate eines Skripts zu speichern oder zu bearbeiten.

2.5 zipLib

2.5.1 Allgemein

ZipLib ist eine in PHP geschriebene Klasse, die das zippen von einzelnen Dateien sowie ganzen Ordnern erlaubt. Um diese zu nutzen, muss sie wie die PHP Flickr Klasse einfach eingebunden werden.

2.5.2 Beispiel

```
<?php
    include 'zipLib/zip.lib.php';

    $zipfile = new zipfile("Test.zip");
    $zipfile->addDirContent('Test/');
    $zipfile->addFileAndRead('Test.xml');

    file_put_contents("Test.zip", $zipfile->file());
?>
```

Abbildung 2.9: AJAX Handler

Auf dem Bild ist ein kleines Skript zu sehen, welches mit Hilfe von ZipLib eine Zipdatei erstellt. Dazu muss als erstes die `zip.lib.php` Klasse eingebunden werden. Der Ordner "Test" und die Datei `Test.xml` werden zu dem Archiv hinzugefügt. Anschliessend wird das Archiv unter "Test.zip" gespeichert.

2.6 Notepad++

Alle Dateien dieses Projekts wurden mit Notepad++ geschrieben. Notepad++ ist ein gratis Texteditor der fast alle gängigen Programmiersprachen unterstützt. Ebenfalls verfügt der Editor über Syntaxhervorhebung und Autovervollständigung.

Kapitel 3

Resultat

3.1 Einleitung

In diesem Kapitel wird alles rund um das Programm abgeklärt. Von der Installation bis zum Aufbau des Programms. Es wird nicht der ganze Programmcode erklärt. Dafür wurde in entsprechenden Code immer Kommentare hinterlassen die jegliche Fragen klären sollten.

3.2 Installation

Da es sich bei dem Programm um eine Website handelt muss nichts spezielles installiert werden. Vorausgesetzt werden lediglich ein Webserver. Die Dateien können einfach auf dem Webserver entpackt werden. In der php.ini des Webservers müssen wenige Veränderungen gemacht werden, diese werden im späteren Abschnitt 3.7.3 erklärt.

3.3 ProgrammoberfläChr

3.3.1 Suche

The screenshot shows the 'Crawl Flickr' search interface. It features a large title 'Crawl Flickr' in blue and pink. Below the title, there are several input fields and buttons. The interface is annotated with numbers 1 through 11:

- 1: A text input field for the search query.
- 2: A text input field for the number of images, with a label 'Number of images:'.
- 3: Two date input fields labeled 'Start date:' and 'End date:'.
- 4: Two text input fields for the maximum aspect ratio, separated by a slash, with a label 'Max aspect ratio:'.
- 5: A text input field for the subsequent file location, with a label 'subsequent file location:'.
- 6: Radio buttons for 'Tag' (selected) and 'Fulltext'.
- 7: A checkbox for 'Tagmode (and)'.
- 8: A dropdown menu for sorting, currently set to 'date-posted-asc'.
- 9: A 'Search' button.
- 10: A 'Preview' button.
- 11: A label for the 'Preview' button.

1. **Suchtext:** Hier wird der entsprechende Text eingegeben, nachdem gesucht werden soll. Bei mehreren Tags sollten diese einzeln mit einem Komma getrennt werden. Bei einer Volltext Suche können die einzelnen Wörter auch mit einem Leerzeichen getrennt werden.
2. **Bildanzahl:** Es kann ein Wert zwischen 1 und 500 gewählt werden. Da die Flickr API ein Maximum von 500 Bildern pro Suchlauf vorschreibt, darum wurde ein Maximum von 500 Bildern definiert. Falls nichts definiert wird, wird standardmässig nach 20 Bildern gesucht.
3. **Startdatum:** Das Startdatum sollte zusammen mit dem Enddatum gesetzt werden. Einzelne sind nicht gültig. Das Format muss so aussehen 21.04.1987.
4. **Seitenverhältnis:** Das Maximale Seitenverhältnis kann definiert werden, durch Eingabe von X und Y. Maximal bedeutet, dass der grössere Wert durch den kleineren geteilt wird. Das Seitenverhältnis der Bilder wird genau gleich berechnet und wenn der daraus entstandene Wert kleiner oder gleich dem gewünschten parameter ist, ist das Bild gültig. z.B. Wenn 16/9 eingegeben wird sind alle Bilder gültig, deren Verhältnis kleiner gleich 1.7 ist.
5. **Speicherort:** Dieses Programm wurde für die IDIAP entwickelt. Deren XML Struktur enthält den genauen Speicherort der Bilder. Den späteren Speicherort, kann man hier eingeben und dieser wird im XML gespeichert.
6. **Suchmodus:** Die Suche stützt sich entweder auf Tags oder auf den Text. Hier kann man wählen, welche Methode man brauchen möchte. Tag durchsucht nur die Tags der Bilder. Text durchsucht Titel, Beschreibung und Tags.

7. **Tagmode:** Diese Option funktioniert nur wenn die Suche mit Tags durchgeführt wird. Bei eingeschaltetem Tagmode, muss ein Bild alle eingegebenen Tags enthalten. Bei deaktiviertem Tagmode muss nur eines der Tags enthalten sein.
8. **Enddatum:** Das Enddatum hat das gleiche Format wie das Startdatum.
9. **Sortierung:** Es können verschiedene Sortierungsmöglichkeiten gewählt werden. Je nach Sortierung können andere Bilder gefunden werden. falls eine Volltext suche gewählt wurde, wird automatisch die Option *relevance* hinzugefügt. Diese Option ist bei der Tagsuche nicht verfügbar.
10. **Suche:** Startet die Suche und generiert direkt XML und Zip File. Nach Beendigung wird der Download sofort bereitgestellt und kann benutzt werden.
11. **Previewsuche:** Bei dieser Art von Suche wird zuerst eine Vorschau generiert, bei der man einzelne Bilder noch löschen kann. Danach kann man ebenfalls ein XML und Zip file aus seiner Auswahl generieren lassen.

3.3.2 Statuspanel

Das Statuspanel wird standardmässig nicht angezeigt. Es wird erst beim suchen aktiviert. Es zeigt den aktuellen Status der Suche an.

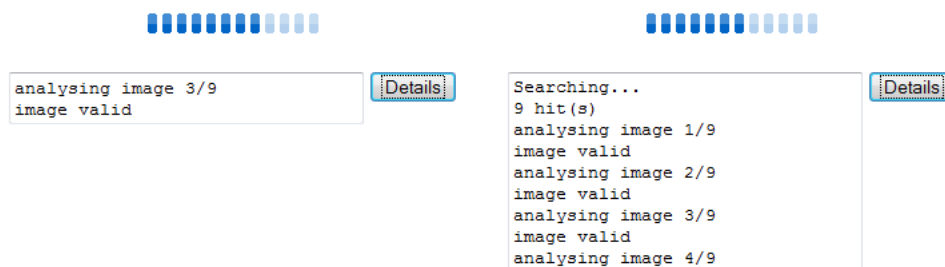


Abbildung 3.1: Statuspanel

Durch einen Button neben des Statuspanels, lässt sich seine Grösse zwischen 1 und 8 Zeilen hin und herschalten. Leider gibt es im Mozilla Firefox keine einzeiligen Textareas, deswegen wird es als zweizeilige Textarea dargestellt.

3.3.3 Bildvorschau

Die Bildvorschau erscheint nach der Analyse des Suchergebnisses. Nachdem die Vorschau geladen ist, wird diese automatisch angezeigt.

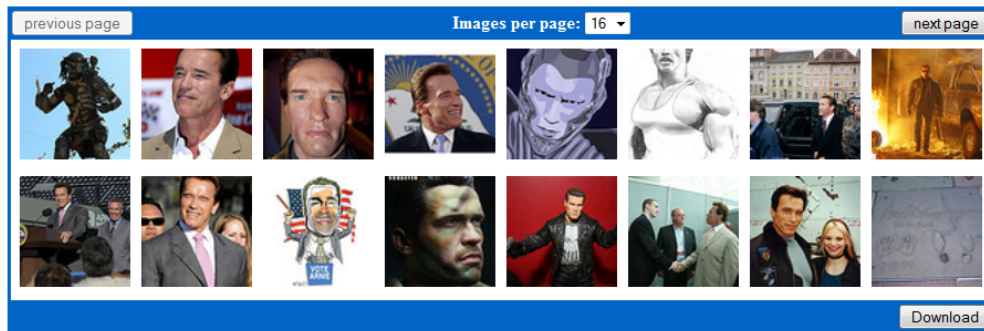


Abbildung 3.2: Bildvorschau

Das Bild zeigt eine solche Vorschau. Oben links und rechts befinden sich Buttons für das vor und zurückblättern der Seiten. Oben in der Mitte gibt es eine Listbox, mit der man die Anzahl Bilder in der Vorschau ändern kann. Damit ein Bild gelöscht wird muss man es einfach anklicken. Nachdem alle Bilder gelöscht wurden, kann man mit dem Button unten rechts ein XML und Zipfile generieren lassen.

3.4 Programmaufbau

3.4.1 Einleitung

Damit später der Ablauf des Programmes genau erklärt werden kann, wird in diesem Teil der Arbeit jedes Skript (php und js), mit all seinen Methoden, genau erklärt. Der Sourcecode jeder Datei ist im Annex enthalten.

3.4.2 search.php

Parameter

- **tag:** Suchtag
- **fulltext:** Volltextsuche (1/0)
- **tagmode:** Tagmode(1/0)
- **sdate:** Startdatum
- **edate:** Enddatum
- **perpage:** Anzahl zu suchender Bilder

- **sort:** Sortierung
- **maxRel:** Seitenverhältnis

Funktion

Dieses Skript arbeitet mit der PHP Flickr API zusammen. Diese wird am Anfang eingebunden. Danach werden alle Parameter überprüft und gespeichert. Diese werden in ein Array gespeichert welches später für die Suche gebraucht werden kann. Als nächstes authentifiziert sich das Skript mit Flickr. Dies geschieht über die eingebundene PHP Flickr API. Ebenfalls über diese API wird die Suche ausgeführt. Das Resultat wird in dem Array *\$photos* gespeichert. Jedes Bild in dem Array wird nun überprüft. Zuerst wird das Seitenverhältnis überprüft. Falls dies in Ordnung ist, wird die Funktion *createArrayItem* ausgeführt. Diese erstellt ein neues Item welches in einem neuen Array *\$validInfos* gespeichert wird. Bei der Erstellung des neuen Items, werden nur wichtige Informationen gespeichert, die später auch gebraucht werden. Falls man zukünftig noch andere Informationen über ein Bild brauchen würde, könnte man in dieser Funktion die entsprechende Information hinzufügen. Nachdem jedes Element bearbeitet wurde wird das Array mittels *json_encode* ausgegeben.

json_encode Gibt eine Zeichenkette zurück, welche die JSON-Darstellung von dem übergebenen Array enthält. Ein Beispiel einer solchen Ausgabe ist auf dem nächsten Bild zu sehen.

```
[{"id":"637859",  
  "title":"the governor.JPG",  
  "description":"The other infamous Austrian born politician.",  
  "flickrUrl":"http://www.flickr.com/photos/48889101687@N01/637859",  
  "picUrl":"http://farm1.static.flickr.com/1/637859_78b4c5a5fb_b.jpg",  
  "thumbUrl":"http://farm1.static.flickr.com/1/637859_78b4c5a5fb_s.jpg"}]
```

Abbildung 3.3: JSON Encode

3.4.3 analyse.php

Parameter

- **url:** Die URL des Bildes, welches gerade analysiert wird.

Funktion

Dieses Skript überprüft die URL eines Bildes. Flickr besitzt für jedes Bild mehrere Grössen. Diese haben jeweils eine spezifische Url. Da die Bilder für dieses Projekt nicht zu klein sein dürfen (min. 400 x 300), sind nur 2 URLs wichtig.

Gross, 1024 an der Längsseite: http://farmfarm-id.static.flickr.com/server-id/id_secret.jpg

Klein, 500 an der Längsseite: http://farmfarm-id.static.flickr.com/server-id/id_secret_b.jpg

Falls eine dieser Url's gültig/verfügbar ist, wird diese ausgegeben. Wenn nicht wird nur ein *false* ausgegeben. Ein wichtiger Teil dieses Skripts ist die Funktion *checkImage()*.

```
function checkImage($url){
    if(!$fp = fopen($url, 'r')){
        return false;
    }

    $meta = stream_get_meta_data($fp);

    if($meta["wrapper_data"][0] == "HTTP/1.1 200 OK"){
        if($exif = @exif_read_data($url, 0, true)){
            if($exif['COMPUTED']['Width'] >= 400 && $exif['COMPUTED']['Height'] >= 300 ||
               $exif['COMPUTED']['Width'] >= 300 && $exif['COMPUTED']['Height'] >= 400){
                return true;
            }
        }
    }
    return false;
}
```

Abbildung 3.4: Url Test

Der Funktion wird eine Url mitgegeben. Mit der Methode *stream_get_meta_data()* werden Informationen über die Url geholt. Danach wird mittels einer If-Abfrage getestet, ob die Zielseite erreichbar ist (*HTTP/1.1 200 OK*). Falls dies der Fall ist, wird in nächsten Schritt die größe des Bildes überprüft. Mit der Methode *exif_read_data()* kann man den Header von Bilddateien auslesen. Dies ermöglicht es, die Grösse des Bildes zu kontrollieren.

3.4.4 getImages.php

Parameter

- **folder:** Der Name des temporären Ordners, wo der Ordner mit den Bildern gespeichert wird. (z.B. */temp/100101/*)
- **picId:** Die ID des Bildes. Dies wird später der Name des Bildes sein.
- **photoUrl:** Die Url des Bildes
- **foldername:** Der Name des Ordners in dem das Bild gespeichert wird. (z.B. *arnold_schwarzenegger*)

Funktion

Am Anfang wird sichergestellt ob der temporäre Ordner schon erstellt wurde. Falls nicht wird dieser erstellt. Danach wird das Bild geladen und gespeichert. Dieses Skript verarbeitet nur ein Bild pro

Aufruf. Dies wurde deshalb so gemacht, damit nicht zu viele Ressourcen des Server verbraucht werden.

3.4.5 createXml.php

Parameter

- **folder:** Der Name des temporären Ordners, wo der Ordner mit den Bildern gespeichert wird. (z.B. */temp/100101/*)
- **id:** Die ID des Bildes.
- **query:** Query der Suche.
- **imgurl:** Die Url des Bildes
- **title:** Titel des Bildes
- **cleantag:** Das Query ohne Leerzeichen und Kommas
- **description:** Beschreibung des Bildes
- **pageurl:** Url zum Webalbum
- **savepath:** Ort wo später die Bilder gespeichert werden.

Funktion

Mit Hilfe dieses Skripts wird ein XML File erstellt. Der Aufbau des XML Files entspricht deren, die in der IDIAP gebraucht werden. Wie in den anderen Skripts wird auch hier jeweils nur ein Bild verarbeiten um Ressourcen zu sparen. Mit Hilfe von DOMDocument kann ein XML Dokument einfach und schnell erstellt werden. Dabei werden nur 2 Methoden gebraucht *createElement()* und *setAttribute()*.

```
$image = $dom->createElement('image');  
$image->setAttribute('id', $id);  
  
$element = $dom->createElement('query',$query);  
$image->appendChild($element);
```

Abbildung 3.5: Url Test

Der kleine Ausschnitt auf dem Bild erstellt ein XML Element mit dem Namen *image*, welches über das Attribute *id* verfügt. z.B. `<image id=1001/>`

3.4.6 zipresult.php

Parameter

- **folder:** Der Name des temporären Ordners, wo der Ordner mit den Bildern gespeichert wird. (z.B. `/temp/100101/`)
- **filename:** Der Name der Zipdatei.

Funktion

Mit Hilfe dieses Skriptes wird eine Zipdatei erstellt. Diese enthält den Ordner, der die ganzen Bilder enthält, sowie die dazugehörige XML Datei. Dazu wird die zipLib benutzt, die in dem Kapitel 2.5 vorgestellt wurde. Wichtig in diesem Skript ist der Befehl `set_time_limit(0);`. Durch ihn wird das Zeitlimit ausgeschaltet. So können auch grosse Resultate gezippt werden, ohne timeout.

3.4.7 crawlHandler.js

Allgemein

Diese Datei koordiniert alle AJAX Aufrufe. Alle PHP Dateien werden von dieser Datei aus aufgerufen. Ebenso wird die Website über diese Datei aktualisiert. Einige der Methoden werden am Ende dieses Abschnitts genauer erklärt.

Variablen

- **photoInformations:** In diesem Array sind alle Bilder und deren Informationen gespeichert.
- **folderName:** Der Name des Ordners, in welchem die Bilder gespeichert werden.
- **directSearch:** Ein Boolean das die Art der Suche definiert. (true = Suche mit anschliessendem Download, false = Suche mit Vorschau)
- **position:** Definiert die Position in einem Array. Wird bei For-Schleifen verwendet.
- **total:** Definiert die totale Anzahl Elemente in einem Array. Wird bei For-Schleifen verwendet.
- **tag:** Suchtag oder Suchtext.

- **cleanTag:** Suchtag oder Suchtext ohne Leerzeichen und Kommas.
- **savePath:** Pfad wo später die Bilder gespeichert werden.
- **previewPage:** Aktuelle Seite der Vorschau.
- **maxImagePerPage:** Maximale Anzahl Bilder pro Vorschauseite.

Methoden

- **createRequestObject:** Gibt ein Requestobjekt zurück, welches für einen Ajax Aufruf gebracht wird.
- **searchHPhotosHandler:** Handler der Funktion *searchPhotos()*. Aktualisiert das Statuspanel, während es die Ergebnisse von *search.php* bearbeitet. Nachdem die Suche abgeschlossen ist wird automatisch die Funktion *startanalysis()* aufgerufen.
- **searchPhotos:** Besitzt den Parameter *searchType*, welcher die Variabel *directSearch* definiert(siehe Variabelbeschreibung). Ausserdem werden alle Variablen aus der Html Datei gelesen und kontrolliert. Wenn alles in Ordnung ist, wird die HTTP Anfrage an *search.php* gesendet.
- **analysisHandler:** Handler der Funktion *analysis()*. Aktualisiert das Statuspanel, während es die Ergebnisse von *analyse.php* bearbeitet. Falls es eine gültiges Ergebnis ist, wird das Bild im Array *photoInformations* gespeichert. Ansonsten wird es gelöscht. Je nach Suchart, wird nach Beendigung der Analyse die Preview oder die Download Funktion gestartet.
- **startanalysis:** Setzt die Position auf 0 zurück und startet die Funktion *analysis()*.
- **analysis:** Sendet eine Anfrage mit den Bildinformationen an die Datei *analyse.php*
- **downloadPhotosHandler:** Handler der Funktion *downloadPhotos()*. Aktualisiert das Statuspanel, während es die Ergebnisse von *getImages.php* bearbeitet. Nachdem der Download fertig ist, wird automatisch die Funktion *startCreateXML()* aufgerufen.
- **startDownloadPhotos:** Setzt die Position auf 0 zurück und startet die Funktion *downloadPhotos()*.
- **downloadPhotos:** Sendet eine Anfrage mit den Bildinformationen an die Datei *getPhotos.php*.
- **createXMLHandler:** Hanlder der Funktion *createXML()*. Aktualisiert das Statuspanel, während es die Ergebnisse von *createXml.php* bearbeitet. Nachdem die XML Datei erstellt wurde, wird automatisch die Funktion *generateZipFile()* aufgerufen.
- **startCreateXML:** Setzt die Position auf 0 zurück und startet die Funktion *createXML()*.

- **createXML:** Sendet eine Anfrage mit den Bildinformationen an die Datei *createXml.php*.
- **generateZipFile:** Da diese Funktion im Gegensatz zu allen anderen nur ein mal ausgeführt wird, ist der Handler in der Funktion enthalten. Diese sendet eine Anfrage mit den Pfadinformationen an die Datei *zipresult.php*. Nachdem die Dateien erfolgreich gezippt wurden, wird automatisch der Download aufgerufen.
- **eraseArrayElement:** Besitzt den Parameter *position*. Dieser gibt die Position des zu löschen- den Elements im Array an. Funktion wird dazu gebraucht ungültige Bilder aus dem Array *photoInformations* zu löschen.
- **changeSearchMode:** Je nach Suchart (Tag oder Text), ändert diese Funktion einzelne Elemente der Website. Dazu gehört Sortierung und die Option Tagmode.
- **previewPictures:** Besitzt die Parameter *actualPage* und *imagePerPage*. Diese Funktion fügt je nach Parameter eine Vorschau zu der Website hinzu. Der erste Parameter *actualPage* definiert dabei welche Seite angezeigt werden soll und *imagePerPage* gibt an wie viele Bilder pro Seite angezeigt werden sollen. Die Funktion erstellt nun für jedes Bild ein IMG Objekt welches der Website hinzugefügt wird.
- **removeImage:** Dieser Funktion wird ein Imageobjekt übergeben. Dies geschieht deshalb, weil diese Funktion nur von den Bildern in der Vorschau aufgerufen wird. Weil das ganze Objekt übergeben wird kann es direkt von der Vorschau und dem Array gelöscht werden.
- **toggleStatusInformations:** Diese Funktion verändert die Grösse des Statuspanels. Wechselt zwischen einer einzeiligen und siebenzeiligen Darstellung.
- **enableLoadGif:** Macht den animierten Ladebalken sichtbar.
- **disableLoadGif:** Macht den animierten Ladebalken unsichtbar.

previewImages

Auf dem folgenden Bild ist ein Ausschnitt aus der Funktion *previewImages()* zu sehen.


```
document.getElementById("previewDiv").style.display="inline";
var parent = document.getElementById('preview');

while ( parent.hasChildNodes() )
{
    parent.removeChild(parent.firstChild);
}

for( i = (maxImagePerPage*previewPage) ; i < max ; ++i)
{
    var id = photoInformations[i]['id'];
    imgObj = document.createElement('IMG');
    imgObj.src=photoInformations[i]['thumbUrl'];
    imgObj.id = i;
    imgObj.setAttribute('style','width:100px;height:100px;margin-top:5px;');
    imgObj.onclick=new Function('removeImage(this)');

    parent.appendChild(imgObj);
}
```

Abbildung 3.6: Preview Pictures

Als erstes wird das *previewDiv* sichtbar gemacht. Danach wird die *parent* Variabel gesetzt. Um sicher zu gehen, dass keine alten Bilder in der Vorschau vorhanden sind, wird jedes dazugehörige Element von dem *parent* Element gelöscht. Dies wird mittels der While-Schleife realisiert. Mit einer For-Schleife werden die einzelnen Bilder nun hinzugefügt. Dazu wird ein IMG-Element erstellt. Danach werden Attribute wie *src*, *id*, *style* und *onclick* hinzugefügt. Danach kann das Objekt dem *parent* Element hinzugefügt werden. Wichtig ist dabei dass in diesem Schritt die onclick-Funktion für das Element gesetzt wird und zwar die Funktion *removeImage()* mit dem Parameter *this*. This bedeutet das Element welches die Funktion aufruft also das Bild selber.

removeImage

Diese Funktion wird ausgeführt falls man auf ein Bild in der Vorschau klickt.

```
function removeImage(image)
{
    var parent = document.getElementById('preview');
    parent.removeChild(image);
    eraseArrayElement(image.id);
    previewImages(previewPage);
}
```

Abbildung 3.7: Preview Pictures

Wie im Bild ersichtlich wird zuerst wieder das *parent* Element definiert. Danach kann das Bild direkt vom Parent gelöscht werden, dazu wir das übergebene Objekt benötigt. Sobald dies gelöscht ist, wird das entsprechende Bild auch noch aus dem Array *photoInformation* gelöscht. Damit die Änderungen auch sichtbar werden, wird am Schluss die Funktion *previewImages()* nochmals aufgerufen. Dies ist auch nötig, da nach der Löschung eines Bildes, die Id's der einzelnen Bilder geändert werden.

3.5 Programmablauf

Da nun alle Klassen und ihre Funktionen bekannt sind, wird in den nachfolgenden Abschnitten der Programmablauf grafisch dargestellt und erklärt.

3.5.1 Bildvorschau

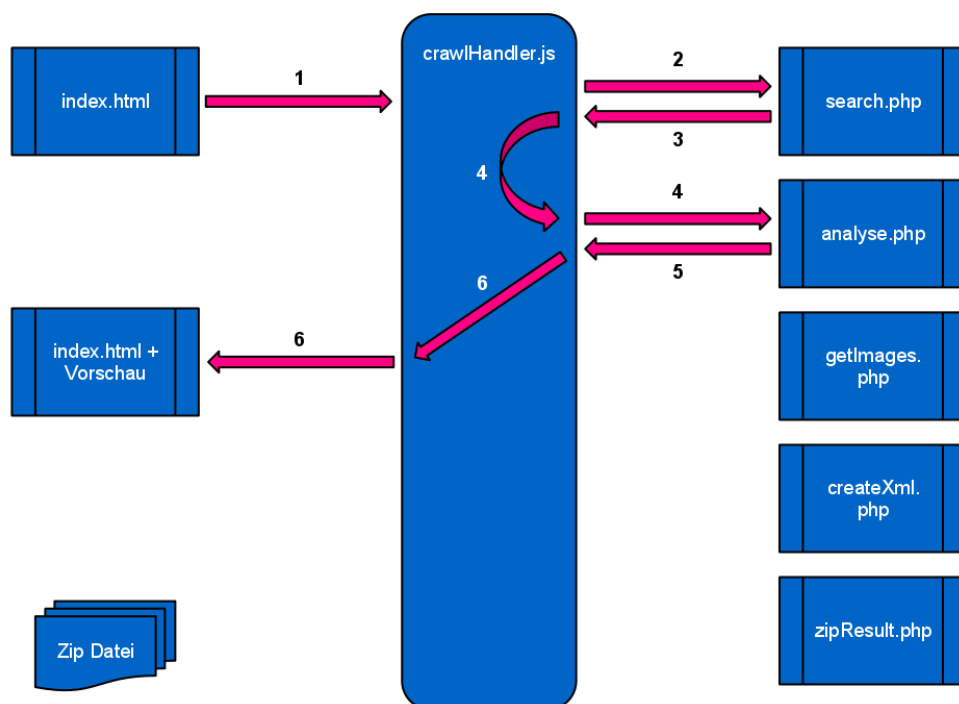


Abbildung 3.8: Programmablauf - Preview

Dieses Bild zeigt den genauen Programmablauf bei der Generierung einer Bildvorschau.

1. Die Attribute aus werden aus dem index gelesen und kontrolliert.
2. Suchanfrage wird gesendet.

3. Antwort wird verarbeitet und gespeichert.
4. Die gefundenen Bilder werden der Analyse übergeben.
5. Gültigen Bilder werden zurückgegeben und gespeichert.
6. Preview wird generiert und der Html Datei hinzugefügt.

3.5.2 Gesamt

Das zweite Ablaufdiagramm beginnt dort wo das vorherige aufgehört hat. Wie man auf dem Bild sehen kann gibt es 2 Wege die zusammenkommen. Einer von der Vorschau und einer von dem Programmablauf (oben). Dies gibt es weil es 2 Arten von Suchen gibt, Suche mit Vorschau und direkte Suche mit Download.

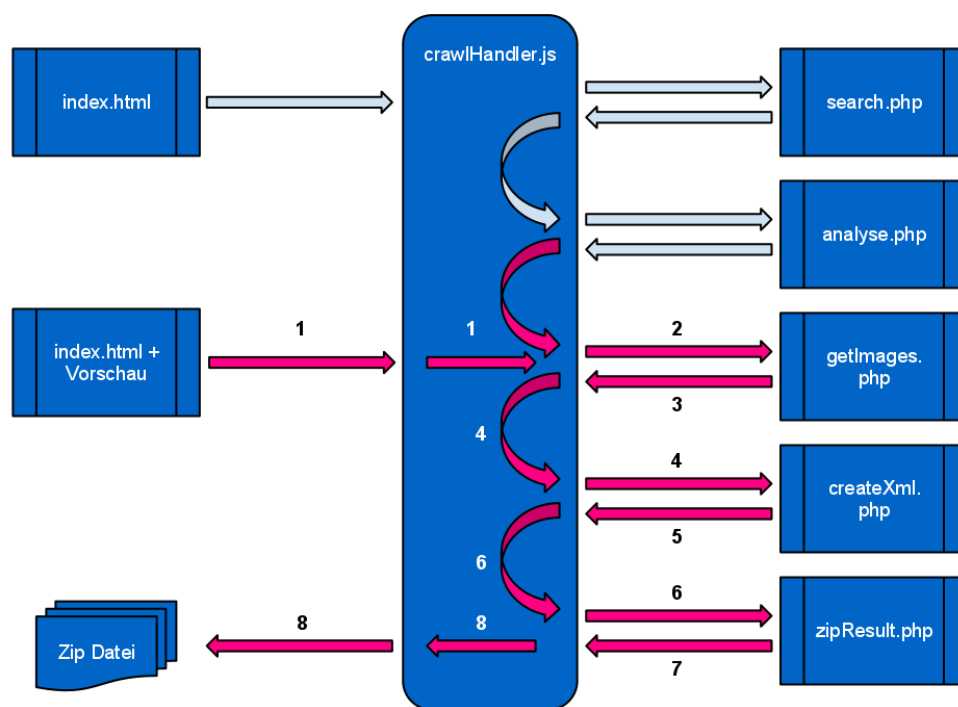


Abbildung 3.9: Programmablauf - Zipdatei

1. Nach der Analyse oder der Vorschau wird direkt an die nächste Funktion weitergeleitet.
2. Links zu den Bildern werden des Skripts übergeben.
3. Skript speichert die Bilder temporär auf dem Server und gibt den Pfad zurück.
4. Bildinformationen werden zur Generierung der XML Datei weitergeleitet.

5. Xml Datei wird in den Temporären Ordner gespeichert.
6. Pfad des temporären Ordners wird an das Skript gesendet.
7. Der Ordner mit den Bildern und der XML Datei werden zu einem Archiv hinzugefügt.
8. Der Link zu der Zipdatei wird an die Website zurückgegeben und automatisch aufgerufen, der download wird automatisch gestartet.

3.6 Test

3.7 Probleme

In diesem Abschnitt werden die Probleme vorgestellt, die während der Arbeit aufgetreten sind. Falls eine Lösung zu dem jeweiligen Problem gefunden wurde, wurde diese ebenfalls aufgeschrieben.

3.7.1 Anzahl Resultate

Nachdem die Bilder gesucht wurden, werden diese analysiert. Da ungültige Bilder gelöscht werden, kann die gewünschte Anzahl Bilder nicht immer garantiert werden. Ein weiteres Problem in diesem Zusammenhang ist, dass die Suche und die Analyse 2 verschiedene Skripts sind. So müsste man nach der Analyse die Suche nochmals ausführen lassen, um die gewünschte Anzahl Resultate zu erhalten.

3.7.2 Löschen temporärer Dateien

Bei jeder Suche mit anschließendem Download, wird ein temporärer Ordner erstellt. Nach mehreren Suchen sammelt sich in dem temporären Ordner viel Bilder, die demnach viel Speicherplatz brauchen. Darum muss von Zeit zu Zeit das *temp* Verzeichnis gelöscht werden. Dies kann leider nicht automatisch erledigt werden, da man nicht kontrollieren kann ob die Datei schon heruntergeladen wurde oder gerade am herunterladen ist. So muss dieser Ordner periodisch manuell gelöscht werden.

3.7.3 php.ini

Da einige Skripts wie *getImages.php* und *zipResult.php* viel Speicher und Zeit beanspruchen, muss man die PHP.ini seines Webservers anpassen.

memory_limit: Legt die Speicherkapazität in MB fest, die ein Skript brauchen darf, bevor der

Parser die Ausführung stoppt. Der Wert sollte auf 300 MB erhöht werden, damit keine Probleme bei der Generierung der Zipdatei entsteht.

3.7.4 Zipdatei

Bei kleinerer Anzahl an Bildern wie 200 ist das Generieren der Zipdatei schnell erledigt. Nimmt man die maximale Anzahl an Bildern (500) scheint die ZiLib Klasse (Kapitel 2.5) Probleme zu haben, diese schnell zu verarbeiten. Da dieses Problem erst gegen Ende der Arbeit gefunden wurde, konnte es nicht dementsprechend behandelt werden.

Kapitel 4

Schlussfolgerung

4.1 Analyse

4.1.1 Planung

Soll Planung

Dies stellt die Soll Planung dar, die bei Projektbeginn erstellt wurde.

Von	Bis	Aufgabe	Zeit in h	
17.05.2010	22.05.2010	Rollup Bachelorarbeit	2	20
		Sitzung mit Experten	1	
		Sitzungsprotokoll erstellen	1	
		Einlesen in Projekt	16	
24.05.2010	29.05.2010	Sitzung mit Experten	1	20
		Sitzungsprotokoll erstellen	1	
		Pflichtenheft erstellen	2	
		Einlesen in Projekt	12	
		Latex Vorlage erstellen	4	
31.05.2010	05.06.2010	Sitzung mit Experten	1	20
		Sitzungsprotokoll erstellen	1	
		Dokumentation	1	
		Planung erstellen	3	
		Einlesen Flickr API	14	
07.06.2010	12.06.2010	Sitzung mit Experten	1	20
		Sitzungsprotokoll erstellen	1	
		Dokumentation	2	
		Einlesen Flickr API	16	
14.06.2010	19.06.2010	Sitzung mit Experten	1	20
		Sitzungsprotokoll erstellen	1	
		Aufbau Testumgebung	3	
		Dokumentation	5	
		Einlesen Flickr Api	5	
		Einlesen Imageanalyse	5	
21.06.2010	26.06.2010	Arbeitsfreie Woche	0	0
28.06.2010	03.07.2010	Arbeitsfreie Woche	0	0
05.07.2010	10.07.2010	Sitzung mit Experten	1	45
		Sitzungsprotokoll erstellen	1	
		Dokumentation	8	
		Entwicklung Imagesuche	35	
12.07.2010	17.07.2010	Sitzung mit Experten	1	45
		Sitzungsprotokoll erstellen	1	
		Dokumentation	8	
		Entwicklung Imagesuche	10	
		Entwicklung Imageanalyse	25	
19.07.2010	24.07.2010	Sitzung mit Experten	1	45
		Sitzungsprotokoll erstellen	1	
		Dokumentation	8	
		Entwicklung Imagesuche	10	
		Entwicklung Imageanalyse	25	
26.07.2010	31.07.2010	Sitzung mit Experten	1	45
		Sitzungsprotokoll erstellen	1	
		Dokumentation	8	
		Entwicklung Interface	25	
		Erste Tests	10	
02.08.2010	07.08.2010	Sitzung mit Experten	1	45
		Sitzungsprotokoll erstellen	1	
		Dokumentation	8	
		Test der Software	20	
		Korrektur	15	

Abbildung 4.1: Soll Planung

Ist Planung

Die nachfolgende Darstellung zeigt die Ist Planung, die wöchentlich aktualisiert wurde.

Von	Bis	Aufgabe	Zeit in h	
17.05.2010	22.05.2010	Rollup Bachelorarbeit	2	20
		Sitzung mit Experten	1	
		Sitzungsprotokoll erstellen	1	
		Einlesen in Projekt	16	
24.05.2010	29.05.2010	Sitzung mit Experten	1	20
		Sitzungsprotokoll erstellen	1	
		Pflichtenheft erstellen	2	
		Einlesen in Projekt	12	
		Latex Vorlage erstellen	4	
31.05.2010	05.06.2010	Sitzung mit Experten	1	20
		Sitzungsprotokoll erstellen	1	
		Dokumentation	1	
		Planung erstellen	3	
		Einlesen Flickr API	14	
07.06.2010	12.06.2010	Sitzung mit Experten	1	20
		Sitzungsprotokoll erstellen	1	
		Dokumentation	2	
		Einlesen Flickr API	16	
14.06.2010	19.06.2010	Sitzung mit Experten	1	8
		Sitzungsprotokoll erstellen	1	
		Aufbau Testumgebung	2	
		Einlesen PHP Flickr	4	
21.06.2010	26.06.2010	Entwickeln Prototyp	10	12
		Sitzung mit Experten	1	
		Sitzungsprotokoll erstellen	1	
28.06.2010	03.07.2010	Einlesen IDIAP Dokumentation	2	2
05.07.2010	10.07.2010	Sitzung mit Experten	1	45
		Sitzungsprotokoll erstellen	1	
		Einlesen JSON	5	
		Neue Latex Vorlage erstellen	5	
		Prototyp weiterentwickeln	20	
		Dokumente weiterschreiben	13	
12.07.2010	17.07.2010	Sitzung/Besuch IDIAP	3	45
		Sitzungsprotokoll erstellen	1	
		Prototyp weiterentwickeln	31	
		Dokumente weiterschreiben	10	
19.07.2010	24.07.2010	Sitzung mit Experten	1	45
		Sitzungsprotokoll erstellen	1	
		Prototyp weiterentwickeln	38	
		Dokumente weiterschreiben	5	
26.07.2010	31.07.2010	Prototyp weiterentwickeln	25	45
		Dokumente weiterschreiben	15	
		Erste Tests	5	
02.08.2010	07.08.2010	Dokumente weiterschreiben	10	45
		Korrektur	20	
		Test der Software	15	

Abbildung 4.2: Ist Planung

Differenzen

Die ersten 5 Wochen der Planung unterscheiden sich nicht schwerwiegend voneinander. Danach wurde hingegen der Planung vermehrt programmiert anstatt die Dokumentation voranzutreiben. Das Problem war, dass in dem Pflichtenheft andere Ziele definiert waren, als die Ziele die man in den Sitzungen mit den IDIAP Mitarbeitern ausgearbeitet hat.

Ebenfalls ein Punkt der zu mehr Programmieraufwand führte war ein Datenverlust nach einem Gewitter. Dieser passierte 4 Wochen vor Abgabe und hatte einige Tage Arbeit als Folge.

4.1.2 Ziele

Da die Ziele im Pflichtenheft ebenfalls in mehrere Kategorien unterteilt waren, werden sie hier in der Zielanalyse ebenfalls unterteilt.

Hinweis: Die schräg geschriebenen Punkte wurden jeweils als optional im Pflichtenheft definiert.

Image Crawling

Zieldefinition	
Imagesuche auf der Onlineplattform Flickr. Dies geschieht über einen Tag, den der Benutzer vor der Suche eingeben kann	<input checked="" type="checkbox"/>
Die Anzahl der Resultate kann vom Benutzer bestimmt werden. Dies wird ebenfalls vor ausführen der Suche vom Benutzer definiert. So kann eine zu grosse Datenmenge verhindert werden.	<input checked="" type="checkbox"/>
Die Standardsuche wird erweiter, damit nicht nur die Tags eines Bildes durchsucht werden, sondern auch Titel, Bildernamen und Kommentare. Es kann gewählt werden welche Kriterien in die Suche mit einbezogen werden sollen.	<input checked="" type="checkbox"/>
<i>Multiple Suche nach verschiedenen Stichwörtern. Diese werden mittels eines Textfiles eingelesen und anschliessend von der Suche verarbeitet.</i>	<input type="checkbox"/>
<i>Imagesuche auf der Onlineplattform Picasa. Hierzu gelten die gleichen Kriterien wie bei der Suche auf Flickr</i>	<input type="checkbox"/>
<i>Neben dem Resultat einer Suche wird zusätzlich ein XML File generiert, das detaillierte Informationen zu dem einzelnen Resultate enthält.</i>	<input checked="" type="checkbox"/>

Abbildung 4.3: Ziele Image Crawling

Beim Image Crawling wurden alle ziele erreicht. Eines der optionalen Ziele wurde ebenfalls realisiert, da dies für die IDIAP von Wichtigkeit war

Bilderanalyse

Zieldefinition	
Ausschliessen von Duplikaten. Durch Analyse von Namen und grösse, können mögliche Duplikate ausgeschlossen werden. So wird die Effizienz der Suche verbessert.	<input type="checkbox"/>
Analyse von Bildparametern. Da bei der Suche nicht alle Parameter der Bilder berücksichtigt werden können, werden die Bilder nach der Suche analysiert. Hier werden Parameter wie Seitenverhältnis und Grösse geprüft. So kann die Bilderliste auf die gewünschten Parameter	<input checked="" type="checkbox"/>
Mittels Algorithmen die die einzelnen Bilder miteinander vergleichen, können falsche Bilder ausgeschlossen werden. Die dazu notwendigen Algorithmen existieren bereits und könnten angepasst und verwendet werden.	<input type="checkbox"/>
Eine Sortierung der Bilder anhand des Farbgehaltes der einzelnen Bilder. Hierzu wird jedes Bild analysiert je nach Farbgehalt sortiert. Bei der Wahl des Farbgehaltes, kann nach mehreren Kriterien sortiert werden z.B von Hell nach Dunkel oder umgekehrt.	<input type="checkbox"/>

Abbildung 4.4: Ziele Bilderanalyse

Bei der Bilderanalyse konnte leider nicht alle Ziele erfüllt werden. Das ausschliessen von Duplikaten wurde im späteren Projektverlauf als zweitrangig eingestuft, da eine Vielfalt von Parametern wichtiger ist.

Webinterface

Zieldefinition	
Webinterface, das auf allen Browsern sowie Betriebssystemen unterstützt wird.	<input checked="" type="checkbox"/>
Verschiedene Felder, für die erweiterte Suche müssen implementiert werden.	<input checked="" type="checkbox"/>
Die Suchergebnisse können als Zip File direkt von der Website heruntergeladen werden.	<input checked="" type="checkbox"/>
Eine Grafische Anzeige des Suchergebnisses. Hier sollten alle Bilder der Suche ersichtlich sein. Die einzelnen Bilder können von dem Benutzer gelöscht werden.	<input checked="" type="checkbox"/>
Einbinden von Statistiken in die Grafische Anzeige des Suchergebnisses.	<input type="checkbox"/>
Erweitern der Grafischen Anzeige, damit multiple Suchergebnisse angezeigt werden können. Dies z.B anhand von verschiedenen Tabs ermöglicht.	<input type="checkbox"/>

Abbildung 4.5: Ziele Webinterface

Es wurden alle gesetzten Ziele erreicht. Ebenfalls zu den obligatorischen Zielen wurde noch eine grafische Anzeige entwickelt.

4.2 Zukünftige Arbeit

Es gibt noch einige Teiler der Website die man verbessern oder ausbauen kann.

- **Geschwindigkeit:** Allgemeine Optimierung der Methoden.
- **Zipdatei:** Da das erstellen von Zipdateien zu viel Zeit in Anspruch nimmt, sollte man die bestehende Klasse Ziplib (Kapitel 2.5) überarbeiten oder durch eine effizientere Klasse ersetzen.
- **Anzahl Resultate:** Erhöhung der maximalem Resultate durch mehrere Skriptaufrufe.
- **Picasa:** Integration von Picasa in die Suche

4.2.1 Persönliche Meinung

Dieses Projekt war für mich eine neue und wertvolle Erfahrung. Die Projekte die wir bis jetzt in der Schule realisiert haben, waren entweder kleiner oder wurden in Gruppenarbeit gelöst. Es war eine grosse Herausforderung von Planung bis Realisation alles selber zu erarbeiten.

Die Zusammenarbeit mit meinem Dozenten sowie den Leuten der IDIAP gestaltete sich einfach und angenehm. Durch den engen Email Kontakt konnten die Ziele des Projekts schnell gefunden werden. Sehr interessant fand ich den Besuch bei der IDIAP in Conthey.

Ein weiterer Interessanter Punkt für mich bei diesem Projekt war die Nutzbarkeit. Wenn das Programm den Entwicklern der IDIAP gefällt, wird es in Zukunft gebraucht und weiterentwickelt. Dies war für mich ein grosser Ansporn mein bestes zu geben.

Mit dem Resultat meines Projekts bin ich im Grosen und Ganzen zufrieden. Klar gibt es immer etwas, das man verbessern könnte.

Von den einzelnen Schritte in diesem Projekt, waren es vor allem die Dokumentation mit der ich meine Mühe hatte. Ein Grund dafür ist, dass dies meine erste grosse Arbeit in Latex ist. Doch auch in diesem Punkt konnte ich auf die Unterstützung meines Dozenten vertrauen, der mit Latex viel Erfahrung hat. So konnte das Projekt ohne grössere Schwierigkeiten beendet werden.

Literaturverzeichnis

PHP.NET (2001): *PHP Referenz*, <http://php.net/> (Stand: 13.08.2010)

W3School (1999): *PHP Tutorial by W3School*, <http://www.w3schools.com/php/default.asp>
(Stand: 05.07.2010)

AJAX F1 (2001): *AJAX PHP Tutorial*, <http://www.ajaxf1.com/tutorial/ajax-php.html>
(Stand: 20.07.2010)

W3School (1999): *AJAX Tutorial by W3School*, <http://www.w3schools.com/Ajax/Default.Asp>
(Stand: 12.07.2010)

W3School (1999): *JavaScript Tutorial by W3School*, <http://www.w3schools.com/js/default.asp>
(Stand: 27.07.2010)

Tony Marston (2004): *PHP XML Tutorial*, <http://www.tonymarston.net/php-mysql/dom.html> (Stand: 06.08.2010)

Edward Tanguay (2003): *Erstellung von XML Dateien mittels PHP*,
<http://www.developerfusion.com/code/3944/how-to-create-xml-files/> (Stand: 06.08.2010)

Yahoo (2010): *Flickr API Key*, http://www.flickr.com/services/api/misc.api_keys.html
(Stand: 26.07.2010)

Yahoo (2010): *Flickr API*, <http://www.flickr.com/services/api/> (Stand: 26.07.2010)

Dan Coulter (2010): *PHP Flickr Framework*, <http://phpflickr.com> (Stand: 20.07.2010)

Wikibooks (2010): *Latex Wikibook*, <http://en.wikibooks.org/wiki/LaTeX/Introduction>
(Stand: 13.08.2010)

PHP Classes (2010): *Ziplib PHP Klassen*, <http://www.phpclasses.org/browse/file/3631.html>
(Stand: 26.07.2010)

DON HQ(2010): *Notepad++ Editor*, <http://notepad-plus-plus.org/> (Stand: 21.04.2010)

Anhang A

Annex

A.1 Aufgabenstellung

HES-SO Valais

EE	IG	EST
X	X	

Données du travail de bachelor
Daten der Bachelorarbeit

FO.2.2.02.18.AB
 stt/16/12/2008

Filière / Studiengang : Informatique de gestion

Confidentiel / Vertraulich ☐

Etudiant / Student	Année / Jahr 2010	No TB / Nr. BA
Proposé par / vorgeschlagen von Henning Müller	Lieu d'exécution / Ausführungsort TechnoArk Expert / Experte	

Titre / Titel: <p align="center">Web image crawling and analysis</p>
Description / Beschreibung: <p>A large amount of images with accompanying text captions are available on the Internet via web pages such as Flickr or Picasa. Exploiting the associations between images and text on the web pages can lead to a virtually infinite source of annotations from which to learn visual models without explicit manual intervention.</p> <p>A distinctive feature of this kind of data is that the images come with captions with a varying quality of labels: "clean label", "noisy label" or "no label" at all. This poses a challenging problem to researchers, that are called to learn correspondences between visual and textual information from very large collections of noisy data.</p> <p>The goal of this bachelor work is to collect, annotate and evaluate a database of images with accompanying text captions, collected from the web. Specifically, the tasks will be:</p> <ol style="list-style-type: none"> 1. Implementing an effective webpage crawler to automatically download images and associate text from the web. 2. Analysis of the text information and their quality for annotating the images. 3. Integration of baseline methods on name and face annotation using tools available from IDIAP. <p>The project will be in collaboration with the IDIAP research institute in Martigny and a six months paid internship would be possible in the context of the bachelor thesis.</p> <p>Début en mai ou septembre</p>
Objectifs / Ziele: <ul style="list-style-type: none"> — Analysis of web pages containing images such as Flickr that can be used for an automatic extraction of meta data — Development of a web crawler to download images and associated textual information — Parsing of html of web pages to understand the structure and the links between the information for example text and images — Creation of simple tools to classify the amount of annotation and the quality for each of the images — Integration of existing tools for automatic annotation of the images and an interface to show the results and allow for a simple correction of the proposed annotations. <p>En liens avec les projets BeMeVIS et Khresmoi.</p>

Signature ou visa / Unterschrift oder Visum Resp. de la filière Informatique de gestion Professeur/Dozent: Henning Müller Etudiant/Student:	Délais / Termine Attribution du thème / Ausgabe des Auftrags: 17.05.2010 (ou septembre) Remise du rapport / Abgabe des Schlussberichts: 16.08.2010 (ou novembre) Exposition publique /Ausstellung Bachelorarbeiten
---	--

Rapport reçu le / Schlussbericht erhalten am Visa du secrétariat / Visum des Sekretariats:

A.2 Pflichtenheft



Bachelorarbeit

Student: Jonas Cicognini

Professor: Henning Müller

Web image crawling and analysis Pflichtenheft

Einleitung

Auf Seiten wie Flickr oder Picasa, gibt es eine grosse Anzahl Bilder. Zu jedem Bild gibt es meist einen Text, sei es ein Titel, Kommentar oder der Name des Bildes. Wenn die Bilder korrekt und sinnvoll beschriftet wären, könnte man auf eine fast unendliche Anzahl gut beschrifteter Bilder zurückgreifen.

Diese könnten als Trainingssets für Bilderkennungsalgorithmen verwendet werden, ohne dass eine manuelle Korrektur nötig wäre. Leider sind nicht alle Bilder gut beschriftet und stellen daher für Entwickler, die auf Bilder mit sinnvollem Kontext angewiesen sind, ein Problem dar.

Das Ziel dieser Arbeit ist es, eine Onlinesuche zu erstellen, mit deren Hilfe die Entwickler ein Set von Bildern suchen und herunterladen können. Das Resultat sollte den Anforderungen der Entwickler entsprechen.

Funktionalitäten

Die Funktionalitäten können thematisch in die folgenden drei Bereiche aufgeteilt werden:

1. Image Crawling

Beim Image Crawling, wird das Web nach bestimmten Bildern durchsucht. In diesem Projekt beschränken wir uns auf grosse Plattformen wie Flickr oder Picasa, die eine grosse Anzahl an Bildern enthalten.

Obligatorische Funktionen:

- Imagesuche auf der Onlineplattform Flickr. Dies geschieht über einen Tag, den der Benutzer vor der Suche eingeben kann
- Die Anzahl der Resultate kann vom Benutzer bestimmt werden. Dies wird ebenfalls vor ausführen der Suche vom Benutzer definiert. So kann eine zu grosse Datenmenge verhindert werden.
- Die Standardsuche wird erweitert, damit nicht nur die Tags eines Bildes durchsucht werden, sondern auch Titel, Bildernamen und Kommentare. Es kann gewählt werden welche Kriterien in die Suche mit einbezogen werden sollen.

Optionale Funktionen:

- Multiple Suche nach verschiedenen Stichwörtern. Diese werden mittels eines Textfiles eingelesen und anschliessend von der Suche verarbeitet.

- Imagesuche auf der Onlineplattform Picasa. Hierzu gelten die gleichen Kriterien wie bei der Suche auf Flickr
- Neben dem Resultat einer Suche wird zusätzlich ein XML File generiert, das detaillierte Informationen zu dem einzelnen Resultate enthält.

2. Analyse der Bilder

Nach der Suche der Daten, werden diese nach verschiedenen Kriterien analysiert. So können falsche oder unerwünschte Bilder gefiltert werden.

Obligatorische Funktionen:

- Ausschiessen von Dublikaten. Durch Analyse von Namen und grösse, können mögliche Dublikate ausgeschlossen werden. So wird die Effizienz der Suche verbessert.
- Analyse von Bildparametern. Da bei der Suche nicht alle Parameter der Bilder berücksichtigt werden können, werden die Bilder nach der Suche analysiert. Hier werden Parameter wie Seitenverhältnis und Grösse geprüft. So kann die Bilderliste auf die gewünschten Parameter beschränkt werden.

Optionale Funktionen:

- Mittels Algorithmen die die einzelnen Bilder miteinander vergleichen, können falsche Bilder ausgeschlossen werden. Die dazu notwendigen Algorithmen existieren bereits und könnten angepasst und verwendet werden.
- Eine Sortierung der Bilder anhand des Farbgehaltes der einzelnen Bilder. Hierzu wird jedes Bild analysiert je nach Farbgehalt sortiert. Bei der Wahl des Farbgehaltes, kann nach mehreren Kriterien sortiert werden z.B von Hell nach Dunkel oder umgekehrt.

3. Webinterface

Hier sollte ein übersichtliches und leicht zu bedienendes Webinterface erstellt werden.

Obligatorische Funktionen:

- Webinterface, das auf allen Browsern sowie Betriebssystemen unterstützt wird.
- Verschiedene Felder, für die erweiterte Suche müssen implementiert werden.
- Die Suchergebnisse können als Rar File direkt von der Website heruntergeladen werden.

Optionale Funktionalitäten

- Eine Grafische Anzeige des Suchergebnisses. Hier sollten alle Bilder der Suche ersichtlich sein. Die einzelnen Bilder können von dem Benutzer gelöscht werden.
- Einbinden von Statistiken in die Grafische Anzeige des Suchergebnisses.
- Erweitern der Grafischen Anzeige, damit multiple Suchergebnisse angezeigt werden können. Dies z.B anhand von verschiedenen Tabs ermöglicht.

Verwaltung des Projekts

Planung Soll/Ist

Am Anfang des Projekts wird eine Planung über den gesamten Zeitraum des Projekts erstellt. Wöchentlich wird überprüft, wie der aktuelle Stand ist und wo es Probleme gibt. Die effektiv gebrauchte Zeit wird jede Woche festgehalten, damit der Unterschied zwischen Soll und Ist jederzeit ersichtlich ist.

Wöchentliche Sitzung

Jede Woche findet eine Sitzung mit dem Dozenten Henning Müller statt um Probleme, den aktuellen Stand und das weitere Vorgehen zu besprechen.

Jonas Cicognini, 31. Mai 2010

Unterschriften

Dozent:

Student:

.....
(Henning Müller)

.....
(Jonas Cicognini)

A.3 Sitzungsprotokolle

Datum: 19.05.2010

Besprochene Themen

Projektablauf

Besprechung des Allgemeinen Ablaufes mit den wichtigen Daten wie Abgabe und Präsentation

Sitzungen

Es gibt jede Woche eine kurze Sitzung mit dem Experten, um den aktuellen Fortschritt zu besprechen und allfällige Probleme zu lösen.

Dokumentaufbau

Es wurde besprochen wie die Abschlussarbeit aufgeteilt werden könnte. Anstelle von Wochenrapporten werden die wöchentlichen Sitzungen schriftlich protokolliert.

Verwendung Latex

Die Abschlussdokumentation wird in Latex geschrieben.

Imageportale

Kurzes Gespräch über die Portale Flickr und Picasa

Aufgaben für die nächste Woche

- Gedanken über die möglichen Features machen, welche implementiert werden könnten.
- Kleine provisorische Planung erstellen.

Datum: 19.05.2010

Besprochene Themen

Aufbau des Pflichtenhefts

Genauer Aufbau des Pflichtenhefts wurde diskutiert. Ebenfalls wurde bestimmt was enthalten sein muss.

Funktionalitäten

Die von mir aufgeschriebenen Funktionalitäten wurden besprochen und ergänzt.

Planung

Die gemachte Planung wurde kurz besprochen. Der Aufbau der Planung war nicht optimal, es wurde entschieden eine Planung zu machen mit den einzelnen Wochen.

Aufgaben für die nächste Woche

- Fertigstellen des Pflichtenhefts
- Entscheiden welche Techniken gebraucht werden
- Planung beginnen

Datum: 19.05.2010

Besprochene Themen

Pflichtenheft

Pflichtenheft wurde von Herrn Müller unterschrieben.

Planung

Die Endgültige Planung ist bis Ende Woche fertig.

Eingesetzte Techniken

PHP und JAVA werden die 2 Techniken die im Projekt eingesetzt werden. Falls nötig werden andere ebenfalls eingesetzt.

Probleme im Latex

Probleme im Latex Template wurden besprochen und gelöst.

Dropbox

Dropbox wird für gesamte Projekt eingesetzt damit der Experte jederzeit die verschiedenen Dokumente betrachten kann.

Aufgaben für die nächste Woche

- Einlesen in die Materie
- Latex Template weitermachen (Titelseite erstellen)
- API von Flickr studieren

Datum: 09.06.2010

Besprochene Themen

Verteidigungstermin

Mögliche Termine für die Verteidigung wurden gemacht.

Sitzungen

Sitzungstermin der nächsten Wochen.

Prototypenentwicklung

Ein erster Prototyp mit den besprochenen Funktionalitäten sollte in den nächsten Wochen entwickelt werden.

Aufgaben für die nächste Woche

- Api von Flickr studieren
- Latex einarbeiten

Datum: 16.06.2010

Besprochene Themen

Bildergrößen Flickr

Das Problem, dass Flickr nur vordefinierte Bildergrößen bereitstellt wurde diskutiert. Der Kunde wird kontaktiert, um abzuklären welche Größen gebraucht werden.

Format der Bilder

Mit Php kann man die Bilder nur in png oder jpg speichern. Hier muss man ebenfalls den Kunden fragen welches Format gebraucht wird.

Aufgaben für die nächste Woche

- Latex einarbeiten
- Prototyp für nächste Sitzung erstellen
- Copyright von Flickr abklären
- Anotationen der Bilder untersuchen

Datum: 08.07.2010

Besprochene Themen

Latexvorlage

Die neue Latexvorlage wurde besprochen.

Dokumentaufbau

Einzelne Abschnitte der Dokumentation wurden genauer besprochen, damit in der nächsten Woche weitergeschrieben werden kann.

Evaluation

Mögliche Kriterien für die Evaluation wurden besprochen.

Prototyp

Es wurden einige Features besprochen, die der weiterentwickelte Prototyp haben soll.

Besuch IDIAP

Der Besuch bei der IDIAP wurde besprochen, gewünschter Tag wäre Dienstagnachmittag.

Aufgaben für die nächste Woche

- Prototyp verbessern
- Dokument weiterschreiben
- Besuch bei der IDIAP organisieren

Datum: 13.07.2010

Besprochene Themen

Latexvorlage

Über einen neuen Latexeditor wurde gesprochen. (Xemacs)

Dokumentaufbau

Der Aufbau des Kapitels „Methoden“ wurde genau besprochen.

Vorbereitung IDIAP

Wurde kurz miteinander besprochen was alles bei der IDIAP Sitzung besprochen werden soll.

Aufgaben für die nächste Woche

- Prototyp verbessern
- Dokument weiterschreiben

IDIAP Besuch am 13.07.2010

Besprochene Themen

Programm IDIAP

Das Annotationsprogramm der IDIAP wurde uns vorgestellt.

XML Format

Das benötigte XML Format für das Programm der IDIAP wurde besprochen.

Dokumentation

Den Arbeitern der IDIAP war es wichtig, dass die Dokumentation so präzise wie möglich ist.

Fotos

Benötigte Minimalgrösse sowie Speichermenge der Fotos wurden definiert. (Minimum 400 x 300 gross und eine maximale Speicherkapazität von 5MB)

Programmaufbau

Der Aufbau des Programms sollte so einfach wie möglich sein, damit die Leute der IDIAP es weiterentwickeln können.

Aufgaben für die nächste Woche

- Prototyp verbessern
- Dokument weiterschreiben

Datum: 22.07.2010

Besprochene Themen

Prototyp

Es wurde über die Fähigkeiten des Prototyps diskutiert, mögliche Verbesserungen wurden ebenfalls diskutiert.

IDIAP

Abgabedatum für die IDIAP wurde nochmals besprochen, einzelne Themen die für die IDIAP wichtig sind. Abgabetermin in digitaler Form für IDIAP: 13.08.2010

Dokumentation

Inhalt der Dokumentation wurde besprochen.

Aufgaben für die nächste Woche

- Dokumentation
- Skript für Xml Generierung