Handbuch Internet-Suchmaschinen
D. Lewandowski (Hrsg.)

Programmierschnittstellen der kommerziellen Suchmaschinen

Fabio Tosques [a,1], Philipp Mayr [b]
[a] *Universität Salzburg, Institut für Romanistik*
Akademiestraße 24
5020 Salzburg
ftosques@gmail.com

[b] *Humboldt Universität zu Berlin, Institut für Bibliotheks- und Informationswissenschaft*
Unter den Linden 6
10099 Berlin
philipp.mayr@gesis.org

Abstract. Im folgenden Beitrag wollen wir die Programmierschnittstellen (APIs) bzw. Web Services vorstellen, die von den drei kommerziellen Suchmaschinen Google, Yahoo! und MSN/Live Search aktuell angeboten werden. Der Beitrag beschreibt die Funktionsweisen der Suchmaschinen-APIs und zeigt Vor- und Nachteile anhand praktischer Beispiele sowie typische Anwendungsbereiche auf. Die angebotenen Web Services bieten Entwicklern die Möglichkeit, die Daten der Suchmaschinen in eigenen Anwendungen zu verwenden und weiterzuverarbeiten. Es ist somit möglich, eigene Programme zu entwickeln, welche die Ergebnisdaten der Suchmaschinen nutzen. In dem Beitrag demonstrieren und diskutieren wir die Potenziale und Anwendungsbereiche, die in solchen frei zugänglichen Diensten liegen. Dabei sollen auch die Nachteile bzw. Diskrepanzen gegenüber den webbasierten Suchmöglichkeiten erwähnt werden, wobei hier besonders die Suchgeschwindigkeit, die Limits und die Unsicherheiten bezüglich der Verfügbarkeit und der Weiterentwicklung genannt werden muss.

Keywords. Suchmaschinen, API, Schnittstellen, Web Service, Google, Yahoo, MSN

Einleitung

Suchmaschinen sind heute die wichtigsten Instrumente zur Navigation, Orientierung und zum Retrieval im Dokumentenraum WWW. Neben den hochoptimierten und allseits bekannten Suchmaschineninterfaces bzw. Web-Frontends im WWW (z.B. www.google.com, www.yahoo.com, www.msn.com), die täglich mehrere hundert Millionen mal abgefragt werden (vgl. [6]), existieren von den genannten Suchmaschinenbetreibern seit wenigen Jahren Web Service basierte Programmierschnittstellen. Diese

[1] Corresponding author

Suchmaschinen-APIs[2] können für verschiedene Suchtypen in den jeweiligen Indizes der Betreiber genutzt werden.

Der Beitrag stellt die Programmierschnittstellen (APIs) der drei kommerziellen Suchmaschinen Google, Yahoo! und MSN/Live Search vor[3]. Die Web Services bieten Entwicklern die Möglichkeit, die Daten der Suchmaschinen in eigenen Anwendungen zu verwenden und weiterzuverarbeiten. Es ist somit möglich, eigene Programme zu entwickeln, welche die Ergebnisdaten der Suchmaschinen nutzen.

Exemplarisch sollen hier zwei Beispiele genannt werden, die Suchmaschinendaten verarbeiten:

1. Webometrics Ranking of World's Universities[4]: Das Webometrics Ranking listet Universitäts-Websites und Dokumenten-Repositories auf Basis einer Reihe aus dem Web generierter Angaben. In dem Ranking wird u. a. die Anzahl der erfassten Webseiten durch die Suchmaschine Google verarbeitet.
2. Publish or Perish[5] (PoP): PoP ist eine Client-Software, die Google Scholar Ergebnisdaten parst und auf Basis der Zitationsdaten der Treffer Indikatoren berechnet[6].

Interessanterweise beziehen beide Anwendungen mehr oder weniger ungeprüft ihre Daten für die Analysen direkt von den Google Ergebnisseiten[7]. Des Weiteren beziehen beide Projekte die Daten über das Hauptinterface der Suchmaschinen (siehe zu den Problemen dieses Verfahrens folgenden Abschnitt). Die APIs werden in beiden Fällen nicht genutzt. Hauptgründe hierfür sind mit Sicherheit die auch in diesem Beitrag erwähnten Unzulänglichkeiten der APIs bzw. das Fehlen einer API im Fall der Software Publish or Perish für die Google Scholar Suchmaschine.

Auch sonst werden die APIs, sei es wegen der genannten Unzulänglichkeiten oder wegen Unkenntnis, relativ selten genutzt. Dabei haben Calishain/Dornfest [2] schon kurz nach der Veröffentlichung der Google API gezeigt, welche Potenziale in diesen stecken. In einem der zahlreichen Beispiele untersuchten sie beispielsweise, ob bei Google das Ranking der Ergebnisse tatsächlich von der Reihenfolge der eingegebenen Suchwörter abhängt. Mit einem Programm, welches die API nutzt, konnten so relativ einfach die Suchwörter permutiert werden und für jede Kombination eine Suchanfrage an Google gesendet und die Daten ausgewertet werden[8]. Schon bei vier Suchwörtern ergeben sich immerhin 24 Kombinationsmöglichkeiten, deren manuelle Abfrage relativ umständlich wäre, besonders wenn die Suche mit unterschiedlichen Suchwörtern wiederholt werden soll. So übernimmt das Programm zum einen die Abfrage und zum anderen kann es gleichzeitig für die Auswertung der Ergebnisse herangezogen werden.

Wir fokussieren den Beitrag zwar auf die APIs der Suchmaschinen, aber auch die APIs von Amazon, Ebay usw. haben ihre Reize, wie Michael Schili erst kürzlich in

[2] API steht für Application Programming Interface, d.h. eine genau definierte und spezifizierte Schnittstelle, die anderen Programmen zur Verfügung steht. Möchte nun ein Unternehmen oder eine Institution, dass Entwickler kontrolliert auf ihre Daten zugreifen können, so müssen Schnittstellen veröffentlicht werden.

[3] Die von ASK (www.ask.com) angebotene Schnittstelle wurde Anfang 2007 wieder eingestellt.

[4] http://www.webometrics.info/about_rank.html

[5] http://www.harzing.com/resources.htm#/pop.htm

[6] An dieser Stelle muss auf die Unzulänglichkeiten der Google Scholar Daten hingewiesen werden (vgl. [10]).

[7] Persönliche Kommunikation mit Isidro Aguillo (Webometrics Ranking) und Anne-Will Harzing (PoP).

[8] http://141.20.126.11/cgi-bin-gs/reihenfolge.cgi

einem Beitrag für das Linux Magazin[9] zeigte. Das von ihm vorgestellte Programm beobachtet täglich die Preisentwicklung von Produkten bei Amazon, die vom Interessenten ausgewählt wurden und sendet eine Mail, wenn der Preis sinkt.
Die beiden letzten Beispiele verdeutlichen, worauf wir uns bezüglich APIs in diesem Beitrag konzentrieren möchten: Personalisierung, Automatisierung und Datenverarbeitung. Zwar können wir dabei nicht alle heutzutage angebotenen APIs beschreiben, da diese jedoch in der Regel die gleiche Technologie verwenden, können die hier beschriebenen Verfahren recht einfach auf andere APIs übertragen werden.

1. Methoden der Datengewinnung im WWW: screen scraping vs. content scraping

Neben den zentralen Herausforderungen Vollständigkeit und qualitativem Ranking der Inhalte lassen sich aktuell folgende konkrete Ziele der kommerziellen Internetsuchmaschinen nennen (vgl. Review in [6]):

- Behandlung von kurzen und unterspezifizierten Anfragen
- Behandlung von Synonymen und Homonymen
- Vermeidung von Suchmaschinen-Spam
- Ermöglichung von zusätzlichen Filtern für unangemessene Inhalte.

Zum Erreichen dieser Ziele können unter Umständen die APIs beitragen. Die Potenziale der APIs werden unseres Erachtens sowohl von den Betreibern, als auch von den Anwendern weitestgehend unterschätzt. So erwähnt Monika Henzinger in ihrem lesenswerten Review mit keinem Wort die Entwicklungen und Potenziale bei den Suchmaschinen-APIs. Das hängt aber vermutlich auch mit den Anforderungen ab, die ein Review in Science erfüllen muss[10].
Des Weiteren bieten die aktuellen Suchmaschinen-APIs Vorteile, die die WWW-basierte Suche nicht bietet:

- **Anpassbarkeit**: Auf eigene Bedürfnisse zugeschnittene Suchinterfaces können selber erstellt und angepasst werden.
- **Automatisierung**: Abfragen lassen sich automatisieren, was umfangreichere Untersuchungen, die beispielsweise über einen längeren Zeitraum laufen sollen, erleichtert (siehe Zeitreihenanalysen in [8,9]).
- **Datenformat**: Die Ergebnisse sind XML kodiert, d.h. die einzelnen Ergebnisse lassen sich maschinell gut weiterverarbeiten, da die einzelnen Informationen eindeutig getaggt, d.h. in genau definierten Tags eingeschlossen sind und so mit XML-Parsern ausgewertet werden können (*content scraping*).
- **Kombinationsmöglichkeiten**: Eine der Stärken der APIs liegt darin, dass sich die verschiedenen Dienste untereinander kombinieren lassen. Wir werden am Ende des Beitrags einige Möglichkeiten aufzeigen.
- **Keine Werbung**: Auf den Ergebnisseiten erscheint keine Werbung. Dies ist natürlich nur für den Anwender ein Vorteil. Den Betreibern der Suchmaschinen ist dies eher ein Dorn im Auge und sicherlich ein Grund dafür, dass die

[9] Michael Schili: „Geiz ist geil“ (http://perlmeister.com/snapshots/200805/index.html)
[10] Persönliche Kommunikation mit M. Henzinger.

APIs nur halbherzig unterstützt werden. Google beispielsweise verdient einen Großteil des Umsatzes mit dem Platzieren von Werbung, den sog. Google-Ads.

Einige dieser Vorteile sind auch ohne Schnittstellen zu erreichen, z.B. mit dem sog. screen scraping: wir „kratzen" die für uns interessanten Informationen aus den zurückgelieferten Webseiten heraus. Das *screen scraping* hat jedoch Nachteile, die sich besonders bei größeren Untersuchungen negativ auswirken können:

- Die zurück gelieferten Seiten sind meist HTML-kodiert. HTML-Seiten sind zwar für Menschen gut lesbar aber praktisch ungeeignet für die programmgestützte Auswertung, d.h. die Seiten sind für Programme gut darstellbar jedoch semantisch schwer zu interpretieren.
- Auch mit Hilfe von speziellen Modulen bzw. Programmbibliotheken, die das *screen scraping* unterstützen und vereinfachen sollen, ist diese Methode der Datengewinnung recht unzuverlässig, da sich das Layout und damit der Code der Webseiten laufend ändern kann.
- Es existiert i. d. R. keine Beschreibung, wie die Informationen getaggt sind. HTML-Tags beschreiben lediglich das Layout und nicht den Inhalt.
- Daten-Overload: überflüssige Daten, die v.a. fürs Layout bestimmt sind, werden beim *screen scraping* über das Netz geschickt. Diese sind für die Auswertung der Suchergebnisse nutzlos und müssen mühsam herausgefiltert werden. Die Rückgabe auf Basis der Anfrage „informetrics" beim Yahoo Web-Frontend bzw. mit den APIs demonstriert die Unterschiede (vgl. Abbildung 1 und 2): das Web-Frontend liefert ca. 8 KB, die API lediglich ca. 2 KB Daten zurück.

Nicht zu unrecht bezeichnet Lincoln Stein [13] das screen scraping als „mediaval torture" und fasst die Probleme folgendermaßen zusammen:

> „Screen scraping is despised for various reasons. First and foremost, it is brittle. Database managers are always tinkering with the user interface, adding a graphic here, moving a button there, to improve the user experience. Each small change in a popular web page breaks dozens of screen-scraping scripts, causing anguished cries and hair-tearing among the bioinformaticists who depend on those scripts for their research of the wet labs they support. Second, it is unreliable. There is no published documentation of what a data source's web pages are supposed to contain […] Finally, there is massive duplication of effort."

Auch wenn Stein sich in diesem Artikel insbesondere auf den Bereich der Bioinformatik bezieht, sind seine Aussagen bezüglich des screen scrapings auch für andere Bereiche der Datengewinnung im WWW gültig, besonders auf jene des Information Retrieval.

Auch die angebotenen APIs haben Nachteile, die hier zumindest erwähnt werden sollen:

- **Content-Unterschiede**: Wie wir in Mayr & Tosques [8,9] gezeigt haben, unterscheiden sich sowohl die Treffermengen als auch die Zusammensetzung des Suchergebnisses zwischen den Standard- und API-Schnittstellen unter Umständen erheblich (siehe auch Demonstrator in Abbildung 15).

- **Performance & Zuverlässigkeit**: Die Interfaces sind nicht in der gleichen Weise optimiert (vgl. vorheriger Punkt). Die APIs sind nicht das Hauptgeschäft der Suchmaschinen-Betreiber und werden daher technisch nicht in derselben Weise unterstützt (z.B. Verfügbarkeit, Geschwindigkeit).[11]
- Das **Parsen der XML-Dokumente** kann sich je nach Größe der Dokumente und der Komplexität der Auswertungsanforderungen trotzdem recht aufwändig gestalten.
- **Limitierung**: Es ist nur eine bestimmte Anzahl Abfragen pro Tag möglich. Die jeweiligen Limits unterscheiden sich je nach API (cfr. Tabelle **2**).

Die folgenden Screenshots (Abbildung 1 und 2) zeigen exemplarisch den Unterschied zwischen screen scraping und content scraping und verdeutlichen Steins Einschätzung. Zu sehen ist jeweils der erste Treffer (in Abbildung 1 rot umrandet) der Ergebnismenge auf die Anfrage „informetrics“.

```
class="res"><div><h3><a class="yschttl"
href="http://rds.yahoo.com/_ylt=A0geu7JDVSFI9qUA4pFXNyoA;_ylu=X3oDMTEzcWhrdGlyBHNlYwNzcgRwb3MDMQRjb2xvA2FjMgR2dGlkA1BSMDI2Xzcy/S
><b>Informetrics</b> - Home</a></h3></div><div class="abstr">About <b>Informetrics</b>. Contact Us. <b>Informetrics</b> develops
presentation for Brazos Town Center, a Lifestyle <b>...</b></div> <span class=url>www.<b>informetrics.com</b></span>  - <a
href="http://rds.yahoo.com/_ylt=A0geu7JDVSFI9qUA45FXNyoA/SIG=15svdqf9m/EXP=1210230467/**http%3a//216.109.125.130/search/cache%3f
</div></li><li><div class="res"><div><h3><a class="yschttl"
href="http://rds.yahoo.com/_ylt=A0geu7JDVSFI9qUA5JFXNyoA;_ylu=X3oDMTEzM2F1cDhlBHNlYwNzcgRwb3MDMgRjb2xvA2FjMgR2dGlkA1BSMDI2Xzcy/S
><b>Informetrics</b> - Wikipedia, the free encyclopedia</a></h3></div><div class="abstr"><b>Informetrics</b> (or Infometrics) is
<b>...</b> Rousseau, Introduction to <b>Informetrics</b>: Quantitative Methods in <b>...</b></div> <span class=url><b>en.wikiped
href="http://rds.yahoo.com/_ylt=A0geu7JDVSFI9qUA5ZFXNyoA/SIG=169kkt812/EXP=1210230467/**http%3a//216.109.125.130/search/cache%3f
</div></li><li><div class="res"><div><h3><a class="yschttl"
href="http://rds.yahoo.com/_ylt=A0geu7JDVSFI9qUA5pFXNyoA;_ylu=X3oDMTEzdjczc2JnBHNlYwNzcgRwb3MDMwRjb2xvA2FjMgR2dGlkA1BSMDI2Xzcy/S
><b>Informetrics</b> - About <b>Informetrics</b></a></h3></div><div class="abstr">About <b>Informetrics</b>. Contact Us. <b>Info
<b>Informetrics</b>, we're proponents of active and ongoing marketing programs to <b>...</b></div> <span class=url>www.<b>inform
href="http://rds.yahoo.com/_ylt=A0geu7JDVSFI9qUA55FXNyoA/SIG=166af0maa/EXP=1210230467/**http%3a//216.109.125.130/search/cache%3f
```

Abbildung 1. screen scraping Beispiel: Anfrage ‚informetrics‘ bei www.yahoo.com – Ausschnitt aus dem HTML-Kode der Ergebnisseite

```
− <Result>
    <Title>Informetrics - Home</Title>
  − <Summary>
      About Informetrics. Contact Us. Informetrics develops integrated marketing ... See the new website and presentation for Braz
    </Summary>
    <Url>http://www.informetrics.com/</Url>
  − <ClickUrl>
      http://uk.wrs.yahoo.com/_ylt=A9iby4qiVSFIlRIAhw3dmMwF;_ylu=X3oDMTB2b2gzdDdtBGNvbG8DZQRsA1dTMQRwb3MDMQRzZ
    </ClickUrl>
    <DisplayUrl>www.informetrics.com/</DisplayUrl>
    <ModificationDate>1202198400</ModificationDate>
    <MimeType>text/html</MimeType>
  − <Cache>
    − <Url>
        http://uk.wrs.yahoo.com/_ylt=A9iby4qiVSFIlRIAiA3dmMwF;_ylu=X3oDMTBwZTdwbWtkBGNvbG8DZQRwb3MDMQRzZWMDc3Il
      </Url>
      <Size>6144</Size>
    </Cache>
  </Result>
− <Result>
```

Abbildung 2. content scraping Beispiel: Anfrage mit der Yahoo API – Ausschnitt aus dem XML-Kode

2. Web Services – die Technik dahinter

Bevor wir auf die einzelnen Services genauer eingehen, möchten wir kurz die Techniken beschreiben, welche von den drei Anbietern verwendet werden. Die Idee, dass

[11] Die Performance hängt zusätzlich von weiteren Faktoren ab: eigene Rechenleistung, verwendete Programmiersprache, Netzanbindung usw.

Computer untereinander Daten austauschen ist alles andere als neu. Heutige Web Services sind nur die konsequente Weiterentwicklung verschiedener Ideen und Entwicklungsstufen, die mit Suns RPC (Remote Procedure Call) und Microsofts COM (Compoment Object Model), DCOM (Distributed Component Object Model) und CORBA (Common Object Request Broker Architecture) begannen. Während letztere an die Entwickler relativ hohe Anforderungen stellten, sind die inzwischen entwickelten Web Services einfacher zu nutzen und stellen damit eine weitaus niedrigere Einstiegshürde dar.

Als Kerntechnologie heutiger Web Service Entwicklungen ist die Extensible Markup Language (XML) zu nennen. Ein wohlgeformtes und gültiges XML-Dokument alleine reicht jedoch nicht für einen Web Service. Für die Entwicklung eines solchen Services wird zusätzlich eine Technologie benötigt, mit der die Daten zwischen Computern, besser zwischen Client und Server, ausgetauscht werden können. Dieser Austausch muss in standardisierter Form erfolgen, damit der jeweilige Empfänger das gesendete XML-Dokument korrekt interpretieren kann.

Im Folgenden sollen die drei wesentlichen Techniken eingeführt werden, mit denen Web Services, d.h. der Austausch von XML-kodierten Daten, inzwischen realisiert werden. Dem kurzen theoretischen Teil, der die konzeptionellen Grundlagen vermittelt, folgen nach einer Übersicht über die Hauptfeatures der APIs dann die konkreten Realisierungen, die von den drei großen Suchmaschinenbetreibern angeboten werden.

Für die Nutzung der APIs ist die genaue Kenntnis der verwendeten Techniken zwar nicht unbedingt erforderlich, es hilft aber bei der späteren Fehlersuche, wenn es bei den ersten Programmierversuchen zu Problemen mit den eigenen Anwendungen kommen sollte.

2.1. XML-RPC

XML-RPC[12] war die erste konkrete Realisierung eines Web Services, der, wie der Name schon andeutet, RPC-Aufrufe mit XML-Daten über das http-Protokoll realisiert. Es wurde besonders von Dave Winer[13] (Userland Software Inc.) entwickelt und 1998 im Content Management System *Frontier* implementiert. Durch die Offenlegung der Spezifikation sollten andere ermutigt werden, XML-RPC in eigenen Anwendungen zu verwenden.

Die Funktionsweise von XML-RPC ist relativ einfach: der Datenaustausch zwischen Client und Server erfolgt mit XML-kodierten Daten (vgl. **Abbildung 3**).

Zwar nutzt keine der hier beschriebenen APIs XML-RPC, es zeigt sich aber, dass die Weiterentwicklungen wie SOAP/WSDL und REST bezüglich des Datenaustausches ähnlich funktionieren und so XML-RPC konsequent erweitert wurde.

[12] RPC = remote procedure call, d. h. es können Prozeduren auf entfernten Rechnern aufgerufen werden.

[13] Dave Winer war ebenfalls bei der Entwicklung von RSS beteiligt, die Komponente, die den Austausch von Feeds realisiert.

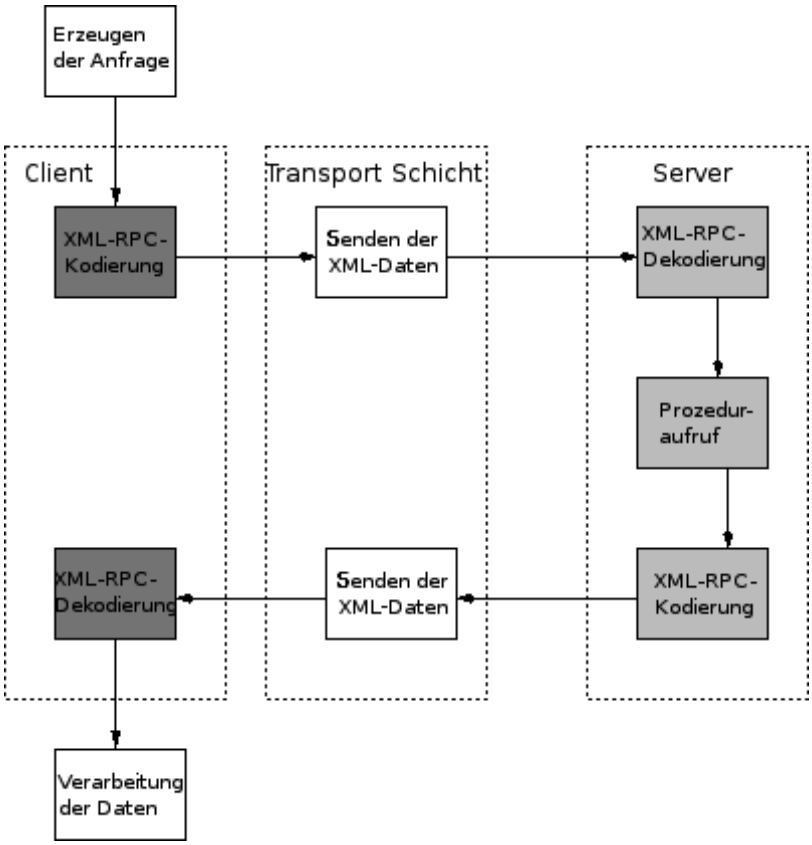


Abbildung 3. Datenaustausch mit XML-RPC

2.2. SOAP/WSDL

Das Simple Object Access Protcol (SOAP) [14] ist die vom W3C entwickelte und empfohlene Spezifikation für die Implementierung von Web Services.

SOAP ist wesentlich flexibler als XML-RPC und bietet ausreichend Raum für individuelle Erweiterungen. Es nutzt viele der intelligenten Lösungen, die XML bietet, wie beispielsweise das Prüfen auf Gültigkeit und Korrektheit der Daten vor dem Senden. Eine SOAP-Nachricht – so werden die Daten bezeichnet, die mittels SOAP-Protokoll zwischen Server und Client übertragen werden – besteht aus:

- Einem Umschlag, dem *envelope-tag*, der die Namespaces deklariert.
- Einem (optionalen) Header, der Informationen darüber enthält, wie der Service die Nachricht verarbeiten soll.
- Einem Textkörper, dem *body-tag*, in dem die Daten enthalten sind und somit die eigentliche Nachricht.

[14] Spezifikation unter: http://www.w3.org/TR/soap/

Die allgemeine Form einer in einen Umschlag verpackten SOAP-Nachricht zeigt Listing 1:

```
<?xml version="1.0" encoding="UTF-8" ?>
<soap-env:Envelope
xmlns:soap-env="http://schemas.xmlsoap.org/soap/envel
ope/">
<soap-env:Header>
<!-- Header-Informationen -->
</soap-env:Header>
<soap-env:Body>
<!-- Daten der Nachricht -->
<!-- Fehlermeldungen -->
</soap-env:Body>
</soap-env:Envelope>
```

Listing 1. Allgemeine Form eines SOAP-Envelopes

Welche Methoden und Variablen der SOAP-Service nutzen und verarbeiten kann, ist in der Web Service Description Language (WSDL) [15] beschrieben. Die WSDL-Metasprache beschreibt die angebotenen Funktionen, Daten, Datentypen und Austauschprotokolle eines Web Service. Die Operationen (Prozeduren/Methoden), die ein Client von außen aufrufen kann, sowie die Parameter und Rückgabewerte sind in derselben WDSL definiert. Abbildung 4 verdeutlicht schematisch, wie der Austausch von SOAP-Nachrichten zwischen Client und Server funktioniert.

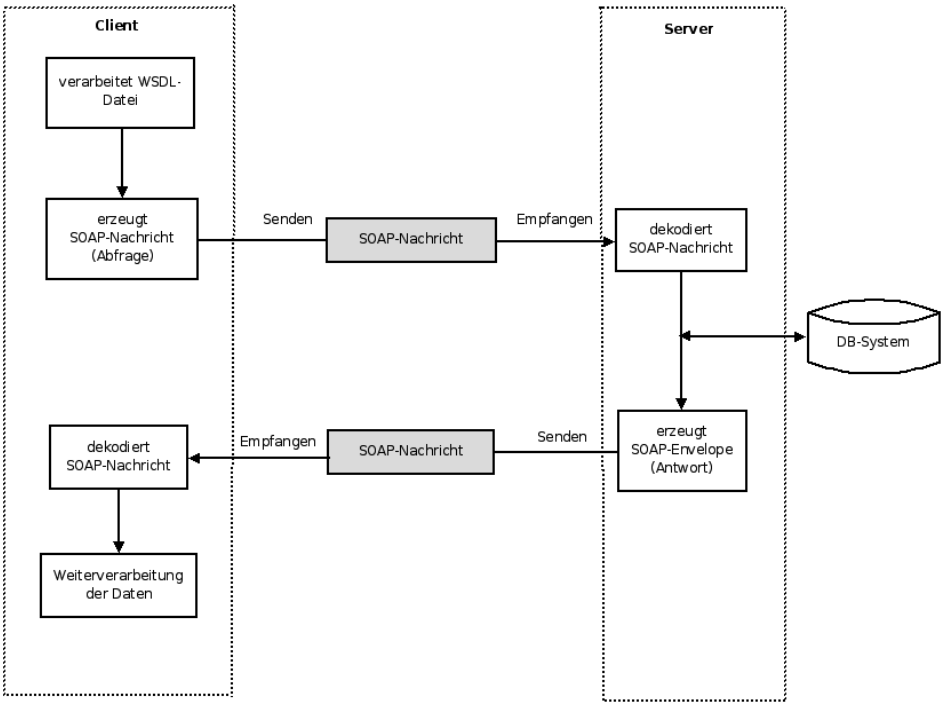


Abbildung 4. Austausch von Nachrichten mittels SOAP/WSDL

[15] Spezifikation unter: http://www.w3.org/TR/wsdl

Von den hier vorgestellten Services nutzen Google und MSN/Live Search das SOAP-Protokoll für den Austausch von Nachrichten.

2.3. REST

Die Web Service Spezifikation Representational State Transfer (REST) wurde von Roy Fielding in seiner im Jahr 2000 veröffentlichten Dissertation entwickelt und vorgestellt [4].

Die Schlüsselinformation im REST-Modell ist die *resource*. Jede Information, die benannt werden kann, ist eine *resource*: ein Dokument, eine Homepage, ein Suchergebnis usw. Ein *resource identifier* identifiziert in der Form eines *uniform resource identifiers* (URI) eine bestimmte Resource, die dann in ihrer Repräsentation betrachtet werden kann: die Webseite selbst, die Repräsentation eines Dokuments usw.
Das klingt zwar zuerst relativ kompliziert, tatsächlich haben aber Web Services, die REST verwenden, die niedrigste Einstiegshürde. So reicht für erste Versuche beispielsweise die einfache Eingabe einer URL (Abbildung 10).

Von den hier vorgestellten Services nutzt einzig Yahoo REST. Die Yahoo Entwickler geben als Hauptgrund für die Favorisierung von REST vor SOAP/WSDL [12] die einfachere Handhabung von REST an:

> „We believe REST has a lower barrier to entry, is easier to use than SOAP, and is entirely sufficient for these services.“[16]

Die Entwickler der Amazon Web Services, auf die wir hier nicht näher eingehen wollen (siehe dazu [15]), bieten beispielsweise beide Möglichkeiten (SOAP/WSDL und REST) an und überlassen die Entscheidung dem Anwender der APIs.

3. Grundlegende Features der Web Services von Google, Yahoo und Live Search

Die folgende Tabelle 1 soll einen ersten Überblick über die grundlegenden Eigenschaften und Unterschiede bezüglich der APIs aufzeigen.

Tabelle 1. Übersicht über die Hauptfeatures der Services

	Google Soap Search API	**Google AJAX Search API**	**MSN Live Search API**	**Yahoo!**
Start	2002 - 2006	2006	2005	2005
Technik	SOAP/WSDL	AJAX	SOAP/WSDL	REST
Status	beta	1.0	1.1	1.2
Registrierung erforderlich	Ja	ja	Ja	ja
Limit (Anzahl der Abfragen pro Tag)	1000	-	25000	5000

[16] http://developer.yahoo.com/faq/#soap

	Google Soap Search API	**Google AJAX Search API**	**MSN Live Search API**	**Yahoo!**
max. Ergebnisse pro Abfrage	10	32[17]	50	100
Developer ID[18]	Ja	nein	Nein	nein
Application ID[19]	nein	ja	Ja	ja
Limit IP-basiert	nein	nein	Ja	ja
Kommerziell nutzbar	nein	nein	ja[20]	auf Anfrage
Leistungsumfang	Websuche, Wortkorrektur, Zugriff auf Cached Pages	Websuche, Blogsuche, Lokale Suche, Videosuche, News, Bildersuche, Buchsuche	Websuche, Wortkorrektur, Cached Pages	Audio, Content Analysis/Term Extraction, Image, Local, MySearch, News, Video, Web, Site Explorer
Beispiele in den Programmiersprachen	Java, C#, VB.NET, Perl	JavaScript, Java	C#, VB.NET, Java, Ruby, Python, Flash	Perl, PHP, Python, Java, JavaScript, Flash

Auf jedes einzelne Feature im Leistungsumfang können wir hier nicht eingehen. Auf den jeweiligen Entwicklerseiten der APIs sind aber ausreichend Beispiele zu den Features vorhanden. Wir möchten uns hier besonders auf die Websuche konzentrieren und die weiteren Suchmöglichkeiten wie Videosuche, Blogsuche, Bildersuche usw., die u.a. von Yahoo und Live Search angeboten werden, weitestgehend außen vor lassen.

4. Google SOAP Search API

4.1. Allgemeines

Die Entwickler von Google waren im Jahr 2002 die ersten, die eine API zur Verfügung stellten, mit der kontrolliert Abfragen an den Google Index durchgeführt werden konnten. Zwar werden seit Ende 2006 die benötigten Schlüssel für Abfragen nicht mehr

[17] Es können maximal die ersten 32 Treffer (jeweils acht auf vier Seiten verteilt) angezeigt werden. Wer mehr als 32 Treffer sehen möchte, wird auf das Web-Frontend von Google weitergeleitet.

[18] Ein Schlüssel (hier: *googlekey*) pro Entwickler (registrierter E-Mail Adresse), der für alle Anwendungen gilt. D.h. ein Schlüssel gilt für alle Programme, die geschrieben werden, und die Anzahl der möglichen Abfragen verteilt sich so auf die jeweiligen Anwendungen.

[19] Für jede Anwendung kann ein eigener Schlüssel (ApplicationID) registriert werden. Dass bei MSN und Yahoo für jede Anwendung eine eigene ID registriert werden kann, ist weit weniger restriktiv, als es bei Google der Fall ist. Die Funktionsweise der applikationsbasierten Limits ist bei Yahoo beschrieben: http://developer.yahoo.com/search/rate.html („Understanding rate limits“ - Zugriff am 02.05.2008).

[20] The service is free to developers for personal use and supports a limited number of calls to the service per day (please see our API terms of use - http://dev.live.com/terms/search.aspx - for details). Commercial users may request higher volumes for daily use, and access to certain types of information unavailable to users of the free service. For more information about commercial use of the service, please send email to api_tou@microsoft.com. (Quelle: http://msdn.microsoft.com/en-us/library/bb251794.aspx).

verteilt, der Dienst selbst ist aber noch verfügbar und funktioniert mit den bis dahin registrierten Schlüsseln. Dieser Status Quo soll laut Google bis auf weiteres erhalten bleiben:

> "This change does not impact current users of the SOAP Search API -- you can continue to execute queries, and we have no plans to turn off the service in the future."21

Auf die Diskussion, ob AJAX tatsächlich die SOAP-API ersetzen kann, gehen wir dann im Absatz Google, AJAX, Search, API ein.

4.2. Was wird für eine Abfrage benötigt und welche Daten lassen sich mit Hilfe der API extrahieren?

Wir werden bei der Google API etwas ausführlicher darauf eingehen, wie ein Programm für eine Abfrage/Auswertung der Ergebnisse aufgebaut ist und welche Elemente vorhanden sein müssen. Da die anderen Services ähnlich funktionieren, können wir uns dort kürzer fassen. Schließlich finden sich auf den Webseiten der Anbieter zahlreiche Beispiele in allen gängigen Programmiersprachen, die den Einstieg erleichtern.

Um herauszufinden, was für eine Abfrage benötigt wird, gibt es zwei Möglichkeiten: entweder nimmt man die Informationen aus der XML-kodierten WSDL-Datei oder (einfacher) aus der vorhandenen Dokumentation[22].

Tabelle 2. Elemente, die für eine Anfrage an den Google Web Service benötigt werden

Schlüsselwort	**Wert**	**Bedeutung**
Key	String: required	Googlekey
Query	String: required	Anfrage
Start	int: min. 0, max 990	Offset des ersten Treffers
maxResults	int: min. 1, max 10	Anzahl der Ergebnisse (nur 1-10 ist möglich)
Filter	true/false	Dubletten aussortieren
Restrict	true/false	Suche auf einen bestimmten Bereich einschränken
safeSearch	true/false	Kindersicherung
Lr	language restricts	Suche auf eine bestimmte Sprache einschränken
Ie	utf-8	input encoding (nur UTF-8 wird unterstützt)
Oe	utf-8	output encoding (nur UTF-8 wird unterstützt)

[21] http://google-code-updates.blogspot.com/2006/12/beyond-soap-search-api.html.
[22] http://code.google.com/apis/soapsearch/reference.html

Eine Anfrage mit der Programmiersprache Perl (siehe Listing 2) hat dann das folgende Aussehen, wobei gut zu erkennen ist, wie die einzelnen Elemente aus
Tabelle **2** verarbeitet werden:

```
   # query google
    my $google_key='[Enter Googlekey]';
    my $google_wsdl = "[Enter path to GoogleSearch.wsdl]";
    my$google_search = SOAP::Lite->service("file:$google_wsdl");
    my $start = 0;
    my $results = $google_search ->
        doGoogleSearch(
                        $google_key,   # der key
                        $query,        # die Abfrage
                        $start,    #    offset    des    ersten    Treffers
# (Zählung beginnt bei 0)
                        10,       # Anzahl der Ergebnisse
                        "true",   # Dubletten aussortieren
                        "",       # restrict (keine Angabe,
                                  # keine Einschränkung)
                        "false",  # safe search
                        "",    # Suche auf bestimmte Sprache
                        "utf-8", "utf-8"  # input/output encoding
              );
```

Listing 2. Anfrage in Perl

Perl erzeugt damit einen „Umschlag“ mit der SOAP-Nachricht, die an den Service geschickt und ausgewertet wird. Dabei muss die Reihenfolge der Elemente eingehalten werden und die obligatorischen Elemente müssen ausgefüllt sein. Sonst erhält man eine Fehlermeldung, einen sog. SOAP-Fault.
Eine Antwort enthält dann die folgenden Elemente:

Tabelle 3. Elemente, die in der Response der Google SOAP API enthalten sind

Schlüsselwort	**Bedeutung**
URL	URL der Ergebnisseite
Snippet	kurze Beschreibung des Ergebnisses
Title	Titel der Ergebnisse
cachedSize	Größe der Seite in KB
relatedInformationPresent	ähnliche Seiten vorhanden?
hostName	Hostname des Ergebnisses
directoryCategory	Eintrag in Google Directory
estimatedTotalResultsCount	Gesamtzahl der Treffer bei Google
searchQuery	Suchanfrage, die geschickt wurde
resultElements	Liste der Resultate
searchTips	Korrekturvorschläge
searchTime	Dauer der Suche (bei Google!)

Wie die Resultate dann mit der Programmiersprache Perl ausgegeben werden können, zeigt das folgende Listing 3, welches der Übersicht halber nur den Titel, die URL und die Zusammenfassung (snippet) der Treffer ausgibt:

```
# Loop through the results.

foreach my $result (@{$results->{resultElements}}) {
   # Print out the main bits of each result
   print
      join "\n",
        $result->{title} || "no title",
        $result->{URL},
        $result->{snippet} || 'no snippet',
        "\n";
}
```

Listing 3. Auswertung der Google Response mit Perl (Ausschnitt)

4.3. Beispiele

4.3.1. Einfaches Beispiel Googly

Fügen wir die beiden oben stehenden Listings zusammen, erhalten wir schon fast ein funktionierendes Perl Programm, mit dem der Google Service abgefragt werden kann. Das komplette Programm ist im folgenden Listing zu sehen. Ein Abfrage kann dann, vorausgesetzt Perl ist installiert, mit perl googly.pl <suchfrage> gestartet werden.

```
#!/usr/local/bin/perl
## googly.pl
# A typical Google Web API Perl script.
## Usage: perl googly.pl <query>
# Your Google API developer's key.
my $google_key='insert key here';
# Location of the GoogleSearch WSDL file.
my $google_wdsl = "./GoogleSearch.wsdl";
use strict;
# Use the SOAP::Lite Perl module.
use SOAP::Lite;
# Take the query from the command line.
my $query = shift @ARGV or die "Usage: perl googly.pl <query>\n";
# Create a new SOAP::Lite instance, feeding it GoogleSearch.wsdl.
my $google_search = SOAP::Lite->service("file:$google_wdsl");
# Query Google.
my $results = $google_search ->
     doGoogleSearch(
           $google_key, $query, 0, 10, "false", "",  "false",
           "", "latin1", "latin1"
     );

# No results?
@{$results->{resultElements}} or exit;
# Loop through the results.
foreach my $result (@{$results->{resultElements}}) {
   # Print out the main bits of each result
```

```
    print
       join "\n",
         $result->{title} || "no title",
         $result->{URL},
         $result->{snippet} || 'no snippet',
         "\n";
}
```

Listing 4. Komplettes Beispiel für eine Abfrage an den Google Service in Perl

Funktioniert alles, sollten die Ergebnisse wie in Abbildung 5 gelistet werden.

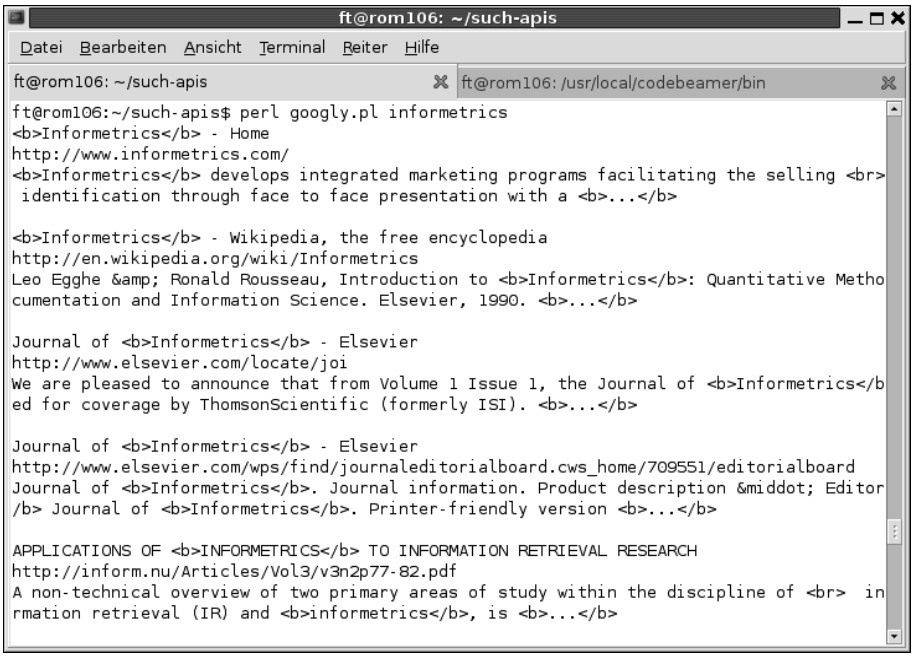


Abbildung 5. Test der Google API mit Perl

Ausgegeben werden hier die Titelzeile, gefolgt von der URL und der kurzen Beschreibung (snippet). Die Experimente mit der Konsole reichen zum Kennenlernen der API völlig aus. Wirklich benutzerfreundlich ist das natürlich nicht, es ist aber relativ einfach, aus dem Perl Programm ein CGI-Skript zu erzeugen, um dem Benutzer ein Frontend für die Eingabe anzubieten[23] (Abbildung 6). Etwas umfangreicher bezüglich der Implementierung ist JGoogly[24], ein auf der Programmiersprache Java/JSP basiertes Userinterface, welches ebenfalls die Google SOAP API benutzt (Abbildung 7).

[23] http://bsd119.ib.hu-berlin.de/cgi-bin-gs/googly.cgi
[24] http://bsd119.ib.hu-berlin.de:8180/JGoogly/

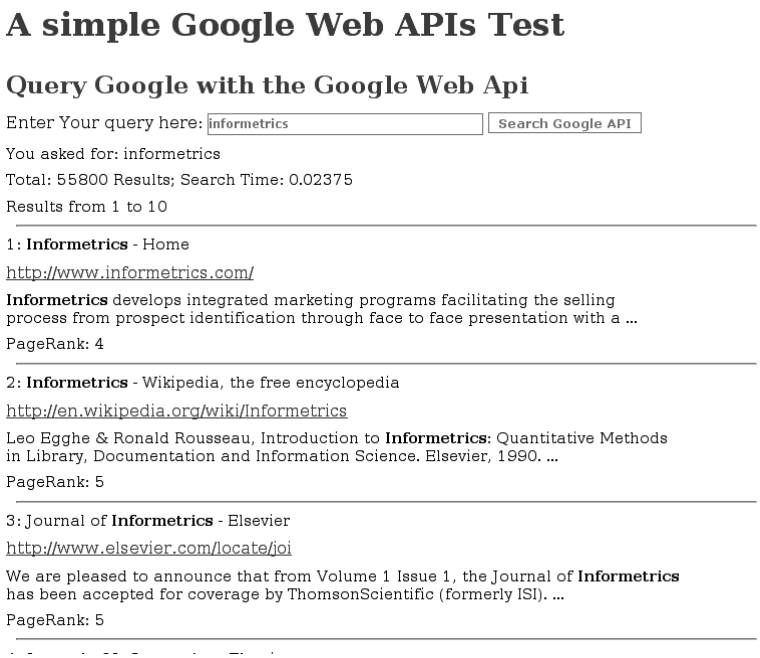


Abbildung 6. Einfaches Google SOAP API Beispiel mit CGI

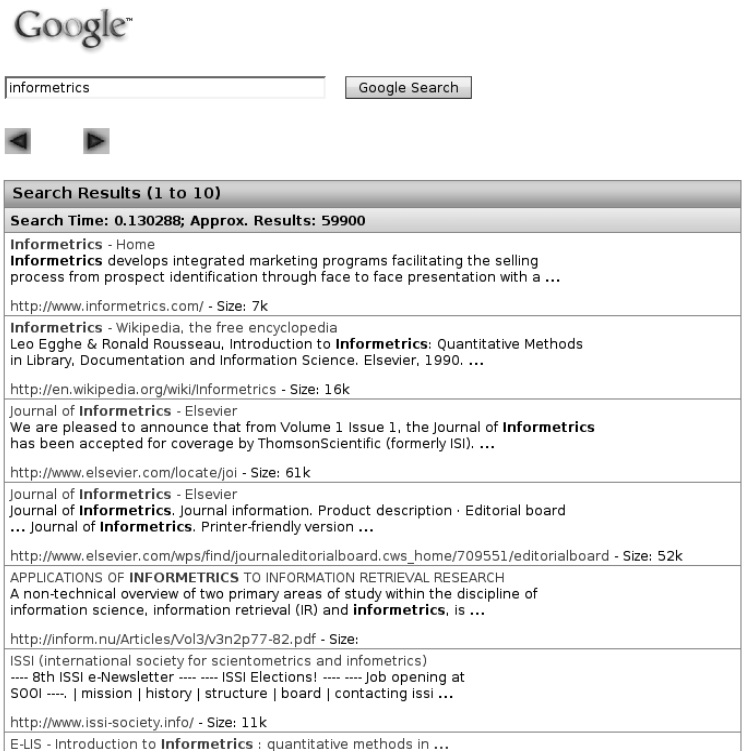


Abbildung 7. Einfaches Beispiel der Google SOAP API mit Java/JSP

4.3.2. Beispiel mit frei definierbarem Output und Anzahl der Ergebnisse

Mit kleinen Änderungen an Googly kann das Programm so erweitert werden, dass es beispielsweise möglich ist, mehr als zehn Ergebnisse (maximal 1000) anzuzeigen oder dem Nutzer die Möglichkeit angeboten wird zu wählen, wie die Ergebnisse aussehen sollen. Dies ist ein wesentlicher Vorteil der APIs, es ist möglich, das Programm auf eigene Bedürfnisse und Präferenzen anzupassen. Die folgende **Abbildung 8** zeigt eine Erweiterung des obigen Beispiels: hier kann der Nutzer wählen, welche Elemente in der Response ausgegeben werden sollen (im Beispiel sind die gewählten Optionen: Titel, URL und PageRank).

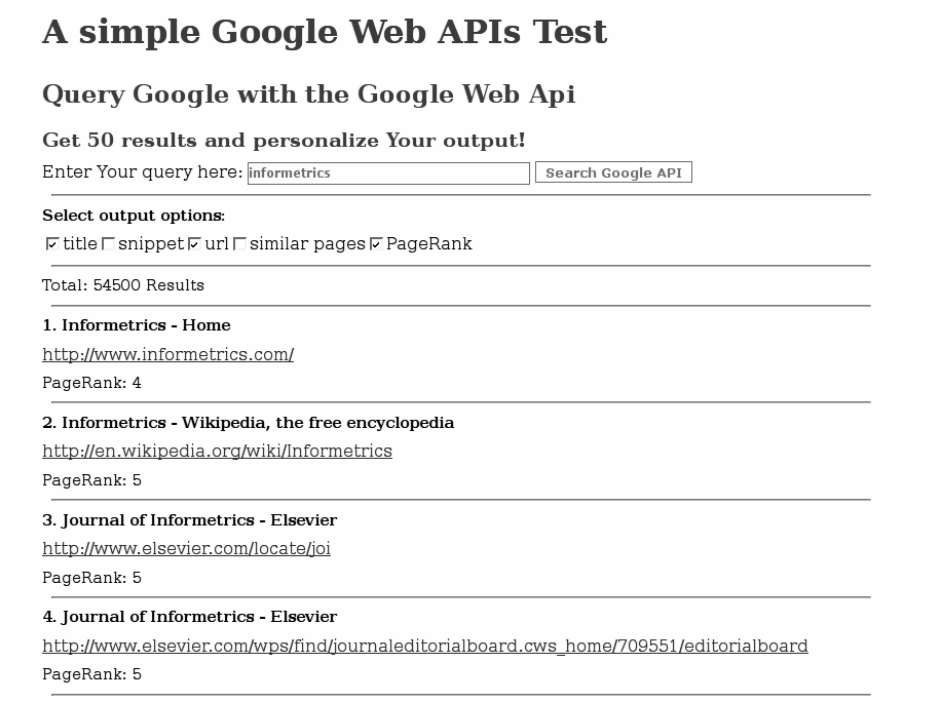


Abbildung 8. Google mit definierbaren Parametern[25]

5. Google, AJAX, Search, API

5.1. Allgemeines

Die Ende 2006 veröffentlichten AJAX[26] APIs können nur schwer mit den SOAP APIs von Google verglichen werden. Zwar bietet die AJAX API mehr Suchmöglichkeiten, wie Blogsuche, Bildersuche, Videosuche usw., jedoch mit der erheblichen Einschränkung, dass pro Suche nur die ersten 32 Ergebnisse angezeigt werden. Die AJAX API ist kein Web Service wie wir ihn in diesem Dokument verstehen, da keine XML-Daten als Response gesendet werden.

[25] http://141.20.126.11/cgi-bin-gs/personal-googly.cgi
[26] AJAX = Asynchronous JavaScript and XML

Die Google AJAX Search API (http://code.google.com/apis/ajaxsearch/) gliedert sich in[27]:

- **Web Search**: damit kann die Websuche, Blogsuche und die Nachrichtensuche in eigene Webseiten integriert werden.
- **Local Search**: damit können Ergebnisse von der lokalen Suche in die eigene Webseite eingebunden werden oder es lassen sich Mash-ups z.B. mit den Google Maps API generieren.
- **Multimedia Search**: damit können Youtube-Videos und Bilder von Googles Bildersuche in die eigene Webseite integriert werden.
- **Book Search**: damit können Ergebnisse von Googles Buchsuche in eigene Webseiten eingebunden werden.

Die Anwendung der AJAX API ist für den Benutzer relativ einfach, es reichen wenige JavaScript Kenntnisse, um die Suchmöglichkeiten in die eigene Webseite einbauen zu können. Daneben bietet Google sogar einen Codegenerator an, für jene, die gar keine Erfahrungen mit JavaScript haben.[28] Damit können erste Experimente durchgeführt werden, die zeigen, wie die AJAX API in die eigene Webseite eingebettet werden kann.

Für die hier beschriebenen Fragen, d.h. wie können mit Hilfe der APIs die Ergebnisdaten weiterverarbeitet und analysiert werden und wie können mit Hilfe der APIs echte Alternativen zu den Web-Frontends der Suchmaschinen angeboten werden, ist die AJAX API jedoch völlig untauglich. Da die Ergebnisse im simplen HTML-Format geliefert werden, ist dies ein Rückschritt: wer die Daten weiterverarbeiten möchte, muss zurück ins „Mittelalter", zum *screen scraping*.[29] Weiterhin ist die strenge Limitierung auf die ersten 32 Ergebnisse ein echtes Hindernis für größere Untersuchungen. Mit der Google SOAP API war es dagegen problemlos möglich, die 1000 Treffer abzufragen, die auch Google anbietet. Damit ist das AJAX Search API kein echter Ersatz für das Google SOAP Search API. In den FAQs zur AJAX Search API fehlt beispielsweise die wichtige Zielgruppe der „researcher" die in den FAQs der SOAP Search API noch ausdrücklich genannt wurden.[30] Die AJAX API ist damit wesentlich restriktiver als es die SOAP API von Google schon war. Dies betrifft zum einen die Gesamtmenge der Treffer und zum anderen das Verbot der Weiterverarbeitung der Daten, wie ein Ausschnitt aus den „terms of use" verdeutlicht:

> "You agree that you will not, and you will not permit your users or other third parties to: (a) modify or replace the text, images, or other content of the Google Search Results, including by (i) changing the order in which the Google Search Results appear, (ii) intermixing Search Results from sources other than Google, or (iii) intermixing other content such that it appears to be part of the Google Search Results; or (b) modify, replace or otherwise disable

[27] Ein Beispiel mit sämtlichen Suchmöglichkeiten ist unter: http://bsd119.ib.hu-berlin.de/~ft/google-ajax-simple-2.html zu finden.

[28] http://code.google.com/apis/ajaxsearch/wizards.html

[29] Dabei sollte jedoch beachtet werden, dass das *screen scraping* ausdrücklich verboten ist: „Can I scrape the search results from the Google AJAX Search API if the API doesn't meet my needs? Sorry, but no; the AJAX Search API is the only permissible way to publish Google AJAX Search API results on your site. We'll block your application if it accesses search results outside of the API." (http://code.google.com/support/bin/answer.py?answer=56502&topic=10021)

[30] "The Google SOAP Search API is for developers and researchers interested in using Google as a resource in their applications." http://code.google.com/apis/soapsearch/api_faq.html#gen1

> the functioning of links to Google or third party websites provided in the Google Search Results."[31]

Einzig am Layout der Ergebnisse dürfen die Nutzer der AJAX API etwas ändern, hier ist Google großzügig.[32] Ansonsten unterliegt die AJAX API ähnlich strengen Regeln wie das Web-Frontend der Suchmaschine.

Entsprechend groß war die Aufregung bei Entwicklern, als bekannt wurde, dass keine neuen Nutzer mehr für die SOAP API akzeptiert werden. Paul Bausch, Autor von Yahoo! Hacks, Amazon Hacks, Flickr Hacks und weiteren Büchern, in denen u.a. auf die jeweiligen APIs eingegangen wird, drückt seine Enttäuschung über Google aus:

> "This is such a bad move because the Google API was **the** [Hervorhebung des Autors] canonical example of how web services work. Not only is Google Hacks based on this API, but hundreds of other books and online examples use the Google API to show how to incorporate content from another site into a 3rd party application."[33]

Über die Gründe für den Wechsel von SOAP zur AJAX API wurde natürlich ebenfalls spekuliert. Die Vermutung von Jason Lefkowitz, Google störe v.a., dass bei der SOAP API die Google Ads fehlen, dürfte einen Kernpunkt treffen:

> "Today, though, Google isn't about search. It's about displaying ads. And in that context, an open API makes no sense — the developer can reformat the search results, and even show them (gasp) without ads!
>
> Hence the "AJAX API", which forces you to take the ads along with the search results. You can't really do much with it, but it does create a new place for Google to show ads on — your blog/site/Web app."[34]

Marc Lucovski von Google nennt im Vortrag über die AJAX API als Gründe für den Wechsel zu AJAX (siehe Fußnote 40):

- die große Anzahl von Entwicklern, die AJAX kennen und nutzen,
- AJAX ist einfacher einzubinden und plattformunabhängig,
- AJAX ist weitaus interaktiver.

Daneben kann als Grund für den Wechsel von SOAP zu AJAX wahrscheinlich auch der Missbrauch durch SEOs[35] vermutet werden, da es mit der SOAP API um einiges einfacher war bzw. ist, die Ergebnisse von Google programmgestützt zu analysieren.

Enttäuscht zeigten sich die Anwender und Entwickler von Google auch deshalb, da sie nicht einmal über die „Einstellung" informiert wurden. Es wäre ein Leichtes für Google gewesen, alle registrierten SOAP API Nutzer per Mail zu informieren. Google hielt es aber offenbar nicht einmal für nötig, Nelson Minar, den Hauptentwickler der SOAP API über die Umstellung zu informieren:

---

31 http://code.google.com/apis/base/terms.html

32 Um die Layoutoptionen zu ändern, können Teile der CSS-Datei von Google (http://www.google.com/uds/css/gsearch.css) überschrieben werden. Dort ist es dann auch möglich, einige Output-Optionen anzugeben, wie z.B. das Unterdrücken von Google-Ads usw.

33 http://radar.oreilly.com/archives/2006/12/google-deprecates-their-soap-s.html

34 http://scripting.wordpress.com/2006/12/19/scripting-news-for-12192006/#comment-25891

35 SEOs steht für Search Engine Optimization, zu Deutsch: Suchmaschinenoptimierung.

> "Man, you leave a company and no one remembers to tell you things. I just learned that two weeks ago Google officially put the SOAP search API on end of life status. That was my first project at Google, as well as one of the last things I worked on before leaving. It looks like the server is still up at least for now, but no new users."[36]

Dass es auch anders geht, zeigte MSN/Live Search, die ebenfalls eine AJAX API anbieten, die SOAP API jedoch nicht für neue Nutzer sperrten und damit beide Zielgruppen im Auge haben: jene, die sich für die Ergebnisse interessieren und mit diesen arbeiten möchten, und jene, die Suchergebnisse auf den eigenen Seiten einbinden möchten.

5.2. Beispiele

5.2.1. Einfaches Beispiel mit der Google AJAX API

Auch wenn die AJAX API für die hier beschriebenen Zwecke wegen der vielen Restriktionen ungeeignet ist, wollen wir kurz ein Beispiel zeigen (**Abbildung 9**). Ein ähnliches Beispiel wird von Google automatisch erzeugt, wenn der Key für die AJAX API registriert wird.[37]

Abbildung 9. Einfaches Google AJAX Beispiel[38]

[36] http://www.somebits.com/weblog/tech/googleSearchAPI.html.
[37] http://code.google.com/apis/ajaxsearch/signup.html
[38] http://bsd119.ib.hu-berlin.de/~ft/google-ajax-simple.html

Weitere Einsatzmöglichkeiten und Funktionen der Google AJAX API können dem Vortrag eines Google Entwicklers entnommen werden, der auf dem *Google Developer Day* am 31. Mai 2007 gehalten wurde: Marc Lucovsky „The Google AJAX Search APIs“[39]. Im Video selbst werden als Zielgruppe für die AJAX Search API besonders Blogger und Web-Designer genannt. Die im Vortrag gezeigten Beispiele verdeutlichen das:

- http://ajaxsearch.blogspot.com/d
- http://www.visualdxhealth.com

Zusammenfassend lässt sich über die Google AJAX Search APIs sagen, dass es sich um eine „nette Spielerei“ handelt. Für weitergehende Untersuchungen der Suchergebnisse sind die AJAX Search APIs v.a. aus drei Gründen nicht zu gebrauchen:

1. Es werden maximal 32 Treffer (8 pro Seite) zurückgeliefert.
2. Die Treffer sind HTML-kodiert und lassen sich nur schwer weiterverarbeiten.
3. Das Anpassen bzw. das Weiterverarbeiten der Ergebnisse ist untersagt.

6. Yahoo! Web Search API

6.1. Allgemeines

Wenn auch mit einiger Verzögerung, zogen Yahoo und MSN drei Jahre nach der Veröffentlichung der Google SOAP API mit eigenen APIs nach. Beide boten jedoch von Anfang an mehr Suchmöglichkeiten als die Google SOAP API (vgl. **Tabelle 2**).

Eine erste Version der Yahoo API wurde Anfang 2005 veröffentlicht.[40] Die Web Search API umfasste von Beginn an neben der Suche im Web die Möglichkeit, nach Bildern, Audios, Videos, nach lokalen Ergebnissen zu suchen und war damit für einen größeren Kreis von Entwicklern interessant als die Google SOAP API.

Die Yahoo APIs sind ausgezeichnet dokumentiert. Ein erster Einstieg in die Websearch API bietet beispielsweise die Seite „Introducing the Yahoo! Web Search APIs“[41]. In der Kategorie „Developer Central“ befinden sich auf der gleichen Seite zahlreiche Beispiele und HowTo Artikel zu allen aktuellen Programmiersprachen.

6.2. Was wird für eine Abfrage benötigt und welche Daten lassen sich mit Hilfe der API extrahieren?

Für eine Suchanfrage müssen dem Service mindestens die Parameter übergeben werden, die in **Tabelle 4** mit „required“ gekennzeichnet sind (appid, query). Alle weiteren Parameter sind optional und müssen nur angepasst werden, falls die default-Werte nicht den eigenen Wünschen entsprechen.

[39] http://www.youtube.com/watch?v=AXgFj_3I_80 (ca. 47 Minuten).
[40] Ankündigung des Services vom Yahoo Entwickler Jeremy Zawodny „Announcing the Yahoo! Search Developer Network and Search Web Services“ (http://www.ysearchblog.com/archives/000084.html)
[41] http://developer.yahoo.com/search/web/

Tabelle 4. Elemente für eine Anfrage an den Yahoo Service

Parameter	**Wert**	**Beschreibung**
Appid	String (required)	ApplicationID
Query	String (required)	Anfrage (UTF-8 kodiert)
Region	string: default *us*	Die regionale Suchmaschine die abgefragt wird, für Deutschland z.B. region=de usw.[42]
Type	*all* (default), *any* oder *phrase*	Art der Suche, die gesendet wird: *all* gibt Resultate mit allen Suchtermen, *any* Resultate mit einem oder mehreren Suchtermen, *phrase* Resultate, in denen die Suchterme als Phrase vorkommen
Results	integer: default 10, max 100	Anzahl der Ergebnisse
Start	integer: default 1	Startposition (Offset) des ersten Treffers
Format	*any* (default), *html, msword, pdf, ppt, rss, txt, xls*	Dateityp, nach dem gesucht werden soll
adult_ok	*no value* (default) oder *1*	Gibt an, ob der Kinderfilter aktiv ist. Eine *1* deaktiviert den Kinderfilter
similar_ok	*no value* (default) oder *1*	Gibt an, ob in den Resultaten ähnliche Seiten erlaubt sind. Eine *1* erlaubt die Suche nach ähnlichen Seiten
Language	string: default *no value* (all languages)	Sprache, in der die Resultatseiten verfasst sind[43]
Country	string: default *no value*	Das Land, auf das die Suche beschränkt werden soll, z.B. für Deutschland: *country=de* usw.[44]
Site	string: default *no value*	Suche auf eine bestimmte Domäne beschränken, z.B. www.yahoo.de. Es könne bis zu 30 verschiedene Domänen in der Suche angegeben werden, z.B. site=www.yahoo.com&site=www.cnn.com
Subscription	string: default *no value*	Subskriptionen zum premium content von Yahoo sollen in die Suche mit einbezogen werden[45]
License	*any* (default), *cc_any, cc_comercial, cc_modifiable*	Dokumente finden, die unter der *Creative Commons licence*[46]stehen.
Output	string: *xml* (default), *json, php*	Dateiformat der Ergebnisse
Callback	string	Name der callback-routine, mit denen die *JSON* Daten umhüllt werden sollen

[42] Unterstützte Regionalcodes: http://developer.yahoo.com/search/regions.html.
[43] Unterstützte Sprachen: http://developer.yahoo.com/search/languages.html.
[44] Unterstützte Länder: http://developer.yahoo.com/search/countries.html.
[45] Subskriptionscodes: http://developer.yahoo.com/search/subscriptions.html.
[46] http://www.creativecommons.org.

Eine Suchanfrage im Browser sähe beispielsweise folgendermaßen aus (hier das Beispiel von http://developer.yahoo.com/search/web/V1/webSearch.html). Gesucht wird nach ‚Madonna' (query=madonna), es sollen nur die ersten zwei Treffer angezeigt werden (results=2)[47].

Das XML-kodierte Ergebnis (Ausschnitt) findet sich in Abbildung 7.

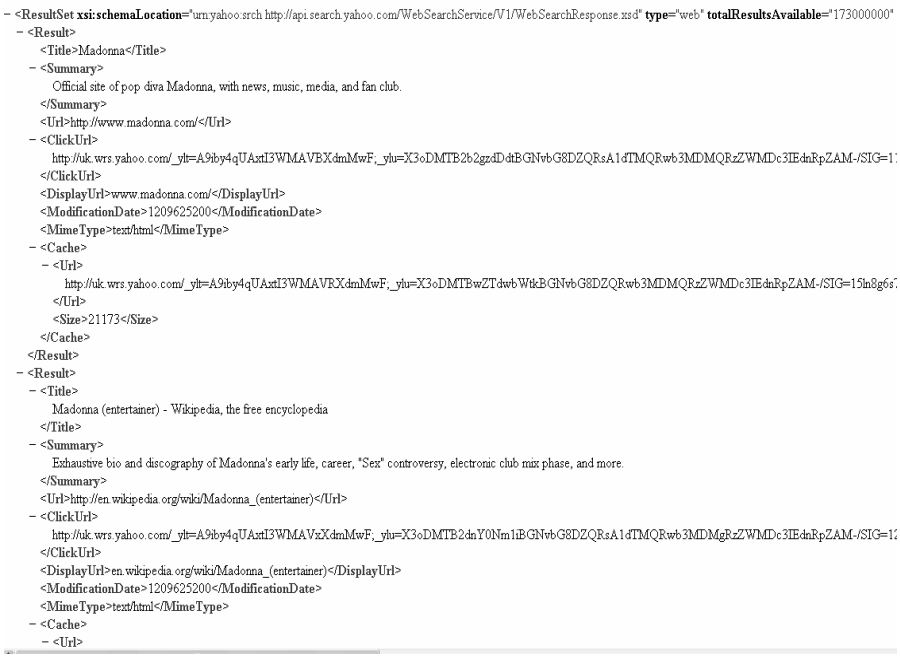


Abbildung 10. XML Response von Yahoo

Diese Anfrage kann nun relativ einfach erweitert werden, z.B.: Suche nach deutschsprachigen Seiten, die unter der CC-Lizenz stehen[48]. Es reicht aus, an die obige URL die Parameter „language=de" und „license=cc_any" jeweils mit dem „&" Zeichen anzuhängen.

Bei diesen einfachen Experimenten zeigt sich die Stärke von REST. Es ist relativ unkompliziert, erste Versuche zu unternehmen und die verschiedenen Parameter auszuprobieren. Die XML-Dokumente im Browser zu betrachten, ist natürlich nur für erste Tests hilfreich. Um eigene Anwendungen zu schreiben, muss das XML-kodierte Ergebnis mit einem XML-Parser weiterverarbeitet werden, damit der getaggte content entweder weiterverarbeitet oder strukturiert ausgegeben werden kann.

[47] http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=YahooDemo&query=madonna&results=2

[48] http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=YahooDemo&query=madonna&results=2&language=de&license=cc_any

Im XML-Dokument, das zurückgeliefert wird, sind die folgenden Elemente enthalten. Diese sind in der XSD-Datei von Yahoo spezifiziert.[49]

Tabelle 5. Elemente, die der Yahoo Service in der Response enthält

Feld	**Beschreibung**
ResultSet	Enthält die Antworten. Attribute sind: totalResultsAvailable: Gesamtzahl der Treffer in der Datenbank bei Yahoo totalResultsReturned: Anzahl der Treffer, die ausgegeben werden firstResultPosition: Position (Offset) des ersten Treffers
Result	Enthält die Elemente jedes einzelnen Treffers
Title	Titel der Webseite
Summary	Zusammenfassung, die mit der Webseite verknüpft ist
Url	URL der Webseite
ClickUrl	URL, um auf die Seite zu verlinken[50]
MimeType	MimeType (~ Dateitpyp) der Seite
ModificationDate	Datum der letzten Änderung der Seite
Cache	Die URL wo die Seite bei Yahoo gespeichert ist und ihre Größe in Bytes

6.3. Beispiele

6.3.1. Einfaches Beispiel mit definierbaren Optionen

Das folgende Beispiel (**Abbildung 11**) ist ähnlich wie das in **Abbildung 8**, in diesem Fall eine Umsetzung mit den Yahoo Search APIs. In der Anfrage wird nach Dokumenten gesucht, die unter einer CC-Lizenz stehen und den Terminus „informetrics“ enthalten. Die Ausgabe der Ergebnisse soll auf den Titel und die URL beschränkt werden. Ähnliche Seiten sollen übersprungen, d.h. von der gleichen Domain sollen maximal zwei Treffer angezeigt werden.

49 http://search.yahooapis.com/WebSearchService/V1/WebSearchResponse.xsd
[50] Zur Bedeutung der ClickUrl: http://developer.yahoo.com/faq/index.html#clickurl

Abbildung 11. Yahoo! mit definierbaren Parametern (Ausschnitt) [51]

7. MSN Live Search API

7.1. Allgemeines

Die MSN/Live Search APIs von Microsoft[52] wurden wie die Yahoo APIs im Jahr 2005 veröffentlicht und boten wie diese von Anfang an mehr Suchmöglichkeiten als die Google SOAP API.

Die Live Search APIs sind vorbildlich dokumentiert und professionell implementiert. Für erste Tests kann die interaktive online Version der SOAP-Version des Search APIs verwendet werden (http://dev.live.com/livesearch/sdk/), siehe **Abbildung 12**.

Die Hilfen zur API sind online bei *MSDN* (Microsoft Developer Network) frei zugänglich.[53] Dort findet sich eine Einführung in die API, eine Referenz und Code Samples. Alternativ kann das *SDK* (Software Development Kit) heruntergeladen werden.[54] [55]

[51] http://141.20.126.11/cgi-bin-gs/personal-yahoo.cgi
[52] http://dev.live.com/livesearch/
[53] http://msdn.microsoft.com/en-us/library/bb264574.aspx
[54] http://www.microsoft.com/downloads/details.aspx?FamilyID=c271309b-02de-42a7-b23e-e19f68667197&DisplayLang=en
[55] Das SDK kommt als Microsoft Installer Paket, es kann also nur unter Windows entpackt werden. Der Installer installiert die Hilfe zur API sowie einige Beispiele z.B. in C:\Programme\Microsoft\Live Search Web Service SDK.

Die Live Search API erlaubt die meisten Abfragen von den drei hier vorgestellten APIs, nämlich 25.000 pro Tag und Applikation[56]. Multipliziert mit den maximal 50 Ergebnissen pro Abfrage ergibt das immerhin 1,25 Millionen Ergebnisse, im Vergleich zu Google mit maximal 10.000 Treffern pro Tag eine deutliche Steigerung. Die WSDL-Datei, in der alle wichtigen Informationen zur Anfrage und zu den Elementen der Resultate stehen, ist für Interessierte verfügbar[57]. Die genaue Funktionsweise und welche Elemente für eine Abfrage bzw. für die Auswertung der Ergebnisse benötigt werden, ist jedoch auch in der Dokumentation ausreichend beschrieben.

7.2. Was wird für eine Abfrage benötigt und welche Daten lassen sich mit Hilfe der API extrahieren?

Für eine Abfrage an Live Search müssen mindestens die folgenden Parameter angegeben werden, die mit „required" gekennzeichnet sind (siehe Tabelle 6):

Tabelle 6. Elemente für eine Anfrage für die Live Search API

Parameter	**Beschreibung**
appId (required)	Bei Live Search registrierte ApplicationID
CultureInfo (required)	Suche auf eine bestimmte Sprache/ein bestimmtes Land beschränken
Flags (optional)	DisableHostCollapsing, DisableSpellCheckFor-SpecialWords, MarkQueryWords
Location (optional)	Angabe von Geokoordinaten
Query (required)	Suchanfrage
Requests (optional)	Angabe, auf welche Quelle zugegriffen werden soll: Image, Web, Phone, Location usw.
SafeSearch (optional)	strict, moderate, off

In den Resultaten befinden sich dann u. a. die folgenden Werte:

Tabelle 7. Elemente, die von der Live Search API geliefert werden

Parameter	**Beschreibung**
CacheUrl	URL der gecachten Seite bei MSN
DateTime	Datum der letzten Änderung der Ergebnisseite
Description	Kurze Beschreibung des Treffers
DisplayUrl	URL der Seite
Summary	Zusammenfassung der Trefferseite
Title	Titel der Trefferseite
Url	URL des Treffers

[56] http://www.viawindowslive.com/LiveSearch.aspx
[57] http://soap.search.live.com/webservices.asmx?wsdl

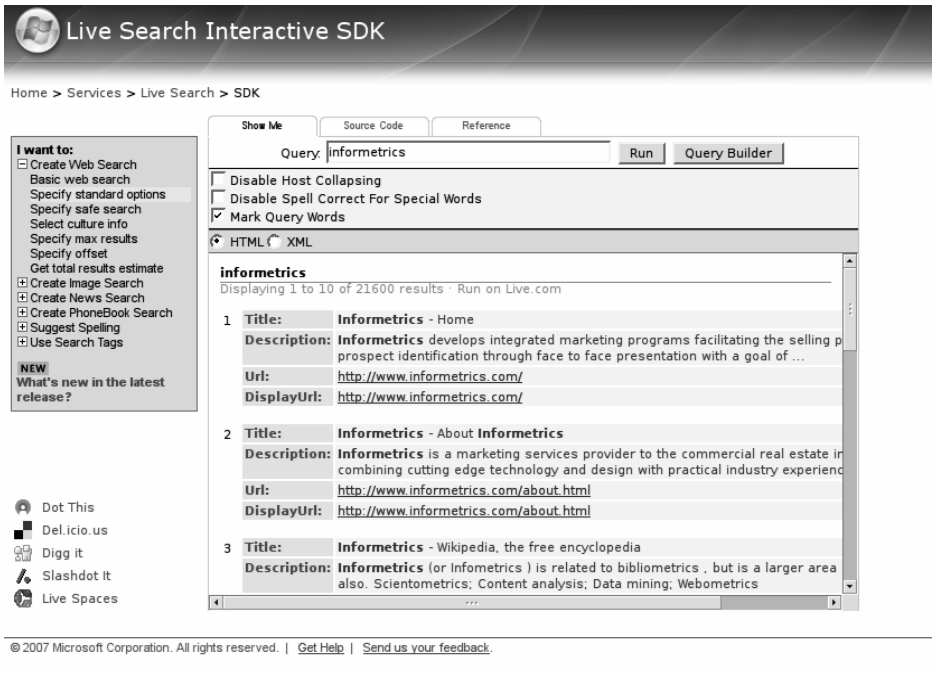


Abbildung 12. LiveSearch interactive (http://dev.live.com/livesearch/sdk/)

Beispiele

In der Live Search Hilfe, bzw. auf den online Hilfeseiten sind für alle denkbaren Suchen und Parameter Beispiele vorhanden. Wir wollen hier kurz ein Beispiel mit konfigurierbaren Optionen vorstellen (Abbildung 13). Das interaktive Beispiel, welches von Live Search angeboten wird[58] (cfr. **Abbildung 12**), zeigt dagegen den vollen Umfang der Möglichkeiten auf, die die API bietet.

8. Anwendungsbeispiele

8.1. Kombinierte Beispiele der einzelnen Services

Schon bei den letzten Beispielen von Yahoo und Live Search haben wir in die Ergebnisseiten den Google Pagerank[59] der Treffer eingebaut und damit eine einfache Möglichkeit gezeigt, wie die Ergebnisse der APIs mit fremden Daten kombiniert werden können. Die APIs können selbstverständlich auch untereinander kombiniert werden, um so beispielsweise einen schnellen Überblick über die ersten 10 Treffer der drei Suchmaschinen zu erhalten (cfr. **Abbildung 14**).

[58] http://dev.live.com/livesearch/sdk/

[59] Für die Abfrage des Google Pageranks verwenden wir das Perl Modul WWW::Google::Pagerank von Yuri Karaban (http://search.cpan.org/~ykar/WWW-Google-PageRank-0.13/lib/WWW/Google/PageRank.pm). Für andere Programmiersprachen finden sich relativ leicht Module oder Klassen, die ebenfalls in der Lage sind, den Pagerank abzufragen, z.B. für Java: http://www.temesoft.com/google-pagerank-api.jsp.

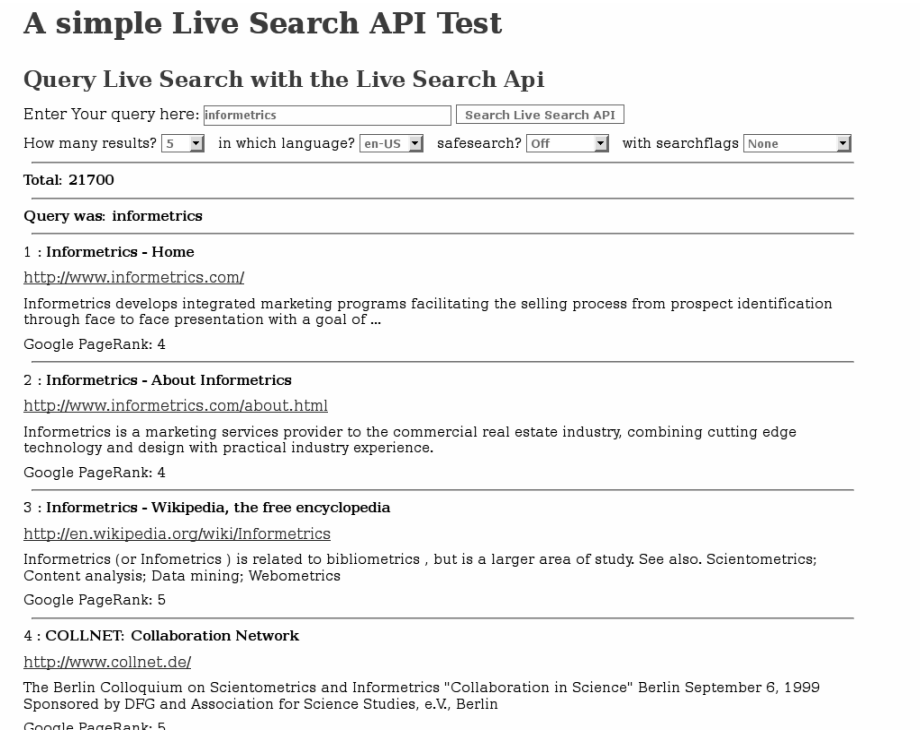


Abbildung 13. Einfaches Live Search Beispiel mit input und output Optionen[60]

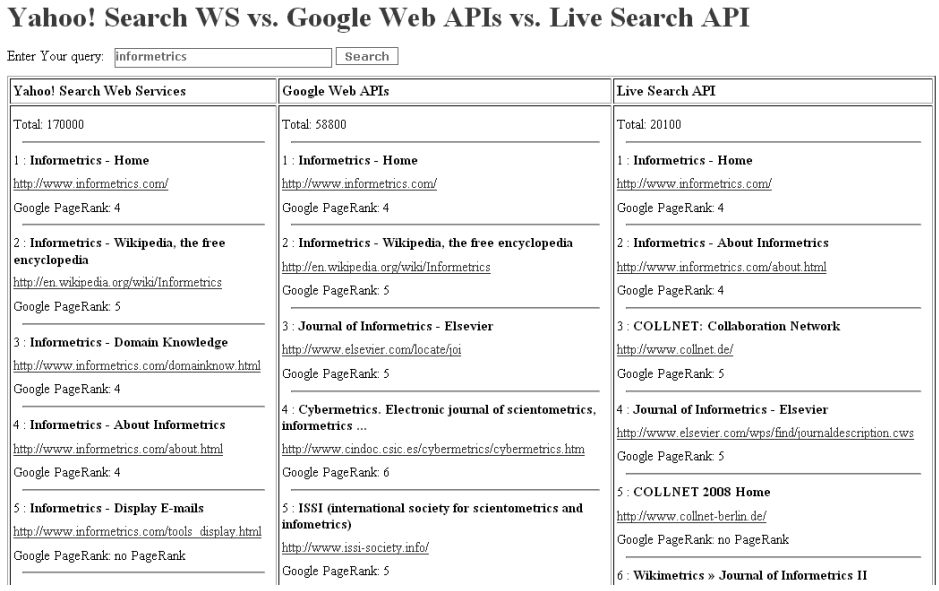


Abbildung 14. Vergleich der drei Suchmaschinen mit den APIs[61]

[60] http://bsd119.ib.hu-berlin.de/cgi-bin-gs/livesearchsimple.cgi
[61] http://bsd119.ib.hu-berlin.de/cgi-bin-gs/yahoo_google_msn_api.cgi

Im nächsten Beispiel vergleichen wir den Output der Google SOAP API mit dem Output des Web-Frontends (**Abbildung 15**). Eine solche Anwendung kann beispielsweise genutzt werden, um zu prüfen, inwiefern sich das Ranking der beiden Schnittstellen von Google unterscheidet.

Die Ergebnisse der einzelnen Zeitreihenuntersuchungen in Mayr & Tosques [8,9] zeigen deutlich, dass die Trefferdaten der beiden Google-Schnittstellen quantitativ sehr ähnlich, aber auf unterschiedlichen Niveaus verlaufen. Starke Schwankungen in den API-Trefferdaten finden sich auch in den entsprechenden Google Web-Treffern. Damit wird deutlich, dass die beiden Treffermengen eng miteinander verbunden sind. Es kann aber nicht davon ausgegangen werden, dass Anfragen an die Google API direkt an die aktuellste Version des Google Gesamtindex (Google Web) weitergegeben werden. Die Trefferdaten der Google API liefern, quantitativ gesehen, nur etwas über 40% der Google Web-Treffer, was darauf hindeutet, dass im Falle der Google API auf einen kleineren und weniger aktuellen Stand des Google-Indexes zugegriffen wird. Dieses Ergebnis der Untersuchung aus dem Jahr 2005 müssen wir inzwischen leicht revidieren. Die beiden Schnittstellen haben sich bezüglich Dokumentmengen weitestgehend angenähert. Zusammenfassend kann man sagen, dass die beiden Suchschnittstellen von Google sich zwar in großen Teilen überlappen, von einer identischen Treffermenge aber nach wie vor nicht ausgegangen werden kann.

Abbildung 15. Vergleich Google SOAP API mit Googles Web-Frontend[62]

[62] http://141.20.126.11/cgi-bin-gs/googleapi_google.cgi bzw http://141.20.126.11/cgi-bin-gs/yahooapi_yahoo.cgi für den Vergleich der Yahoo Schnittstellen.

"First of all it has to be clear that querying the Google APIs does not deliver the same result data as the highly optimized Google Standard interface."
(vgl. [9])

8.2. Datenverarbeitende Beispiele

Die Weiterverarbeitung der Daten, die von den Services pro Abfrage dem Clienten gesendet werden, zeigt die wahren Stärken, die in den alternativen APIs liegen. Hier wird es auch für Forscher interessant, da damit eigene Algorithmen auf die Daten angewendet werden können, um neue Resultate zu erhalten. Da die Resultate eindeutig getaggt, d.h. in genau definierten XML-Tags eingeschlossen sind, können mit Parsern nur die Daten herausgezogen werden, die für eine bestimmte Untersuchung interessant sind (content scraping).

So wäre es denkbar, dass die drei beschriebenen Dienste über einen längeren Zeitraum täglich einmal abgefragt werden, wobei nur die Gesamtmenge der Treffer interessant ist. Diese kann dann sofort in einer Excel-Tabelle oder einer CSV-Datei[63] gespeichert werden, um einfache statistische Analysen durchzuführen. Damit könnten dann Aussagen über die tatsächliche Bedeutung oder die Varianz solcher Gesamttreffermengen getroffen werden.[64]

Wir haben beispielsweise die Top-Level-Domains (TLD) oder die verschiedenen Dateitypen von Suchergebnissen untersucht. Zu letzterem konnte sehr deutlich gezeigt werden, dass die Dokumententypen sich mit der Länge der Suchanfrage ändern, d.h. je präziser die Suchanfrage ist, desto mehr Dokumente vom Typ PDF, DOC, PS usw. befinden sich in der Treffermenge ([7], [8]). Bei der Analyse von TLDs müssen wir auf weitere Module zurückgreifen, wie sie in den bekannten Programmiersprachen vorhanden sind. Zusätzlich zu den URLs aus dem XML-Dokument mit einem XML-Parser müssen aus den URLs die TLDs herausgefiltert werden. Mit *screen scraping* stoßen wir beim Herausfinden der URLs auf größere Probleme, da in den webbasierten Ergebnisseiten jede Menge URLs sind, die nichts mit den eigentlichen Treffern zu tun haben, z.B. Werbung, Verweise auf andere Suchmöglichkeiten, interne Links usw. (vgl. auch [10]).

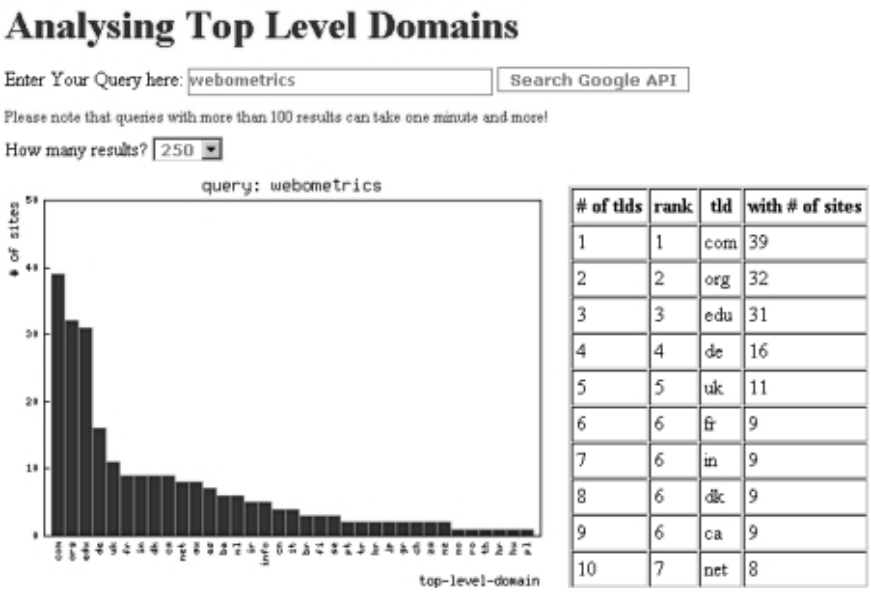
Der folgende Screenshot (**Abbildung 16**) zeigt beispielsweise, wie die Auswertung der TLDs on-the-fly realisiert werden kann und die Daten im Anschluss graphisch und tabellarisch verarbeitet werden können.

Erfreulicherweise konnten in Mayr & Tosques [8] Ergebnisse informetrischer Gesetzmäßigkeiten zumindest in Stichproben bestätigt werden. Sowohl die Verteilung der TLDs in Form der typischen Long-Tail-Distributions als auch der hohe Anteil der PDF-Treffer in der Dateiformat-Analyse konnte in den Treffern der Google API nachgewiesen werden. Damit können die Google Web APIs unseres Erachtens erfolgreich zur Datengenerierung in wissenschaftlichen Internet-Studien eingesetzt werden.

---

[63] CSV = comma seperated values

[64] Eine solche Untersuchung wäre natürlich auch mittels *screen scraping* möglich und zu realisieren, jedoch müsste dann über den gesamten Untersuchungszeitraum immer wieder geprüft werden, ob sich der HTML-Code der Ergebnisseiten nicht ändert und damit das Herauskratzen der interessanten Information nicht mehr funktioniert.



# of tlds	rank	tld	with # of sites
1	1	com	39
2	2	org	32
3	3	edu	31
4	4	de	16
5	5	uk	11
6	6	fr	9
7	6	in	9
8	6	dk	9
9	6	ca	9
10	7	net	8

Abbildung 16. Analyse der Anfrage „webometrics“ nach Top-Level Domains[65]

9. Weitere Einsatzmöglichkeiten

Suchmaschinen-APIs werden zunehmend zur Datenbeschaffung (Data gathering) für wissenschaftliche Studien verwendet (siehe Suche in Google Scholar nach der Phrase „Google API“[66]). Die Google APIs sind in den letzten Jahren zum Beispiel in Forschungsprojekten der Arbeitsgruppen von Thomas Krichel, Mike Thelwall [14], Stevan Harnad [5], Frank McCown [11] oder Steffen Staab [3] eingesetzt worden.

Thomas Krichel[67] benutzte beispielsweise die Google API, um Volltexte von Artikeln, die in DBLP (http://dblp.uni-trier.de) erschlossen sind, zu finden. Sein System sucht Google nach mehreren tausend Titeln. Die Google PDF and MS Word Resultate werden nach Text konvertiert und der Titel gesucht. Google’s HTML wird nach Links zu PDF und Word Dateien abgesucht und die Kandidatenvolltexte aus den Links werden geprüft. Krichel’s System wurde auch in der Zentralbibliothek für Wirtschaftswissenschaften in Kiel für RePEc Volltexte eingesetzt.

Ein anderes interessantes Forschungsprojekt, das laut Heise Meldung [68] vom 04.07.2006 ebenfalls die Google SOAP API für Untersuchungen nutzt, ist der Web-Dienst *docoloc*[69]. Der Dienst soll bei der Suche nach Plagiaten helfen und nutzt neben nicht-öffentlichen Quellen die Google SOAP API.

Kay Kagelmann nutzt hingegen die Yahoo-Schnittstelle, um die Erwähnungen von Prominenten auf Internetseiten zu beobachten.[70] Dabei speichert er einmal wöchentlich

[65] http://141.20.126.11/cgi-bin-gs/tld_chart.cgi
[66] http://scholar.google.com/scholar?as_q=&as_epq=Google+api
[67] Nach eigenen Angaben.
[68] http://www.heise.de/newsticker/Braunschweiger-Software-findet-Plagiate--/meldung/75035
[69] http://www.docoloc.de/
[70] http://www.promitrend.de/

die Gesamttreffer zu einem Prominenten in einer Datenbank und erstellt daraus ein Ranking sowie Statistiken, welche die Veränderungen zu einer bestimmten Person aufzeigen.

Eine Beispielanwendung der Yahoo Image Search API wurde 2005 von Michael Schili vorgestellt.[71] Zu einem Suchbegriff findet das Programm eine Reihe passender Bilder von Yahoo, die das Skript „`slideshow`"[72] dann im 5-Sekunden-Takt wie in einer Diashow nach und nach im Browser darstellt.

10. Fazit

Mit der hier vorgestellten Einführung und den Beispielen zu den Suchmaschinen-APIs von Google, Yahoo und Live Search wollten wir Ideen und Anregungen geben, wie die APIs genutzt werden können. Web Services, wie die hier beschriebenen, lassen sich inzwischen relativ einfach implementieren und anwenden. Auch muss man sagen, dass sowohl die Dokumentation als auch die Übersichtsartikel auf den jeweiligen API-Seiten, als auch die angebotenen Beispiele den Einstieg erleichtern. Daneben haben Bücher wie Google Hacks [2] und Yahoo Hacks [1] die Services populär gemacht.

Im Beitrag konnten wir nur einen Ausschnitt aus dem Leistungsumfang der APIs zeigen. Besonders die Yahoo APIs und die Live Search APIs bieten viele weitere Möglichkeiten und eignen sich daher gut für Untersuchungen der Treffermengen der jeweiligen Suchmaschine. Uns kam es v.a. darauf an zu zeigen, wie die Kriterien Anpassbarkeit, Automatisierung, Weiterverarbeitung der Daten und Kombinationsmöglichkeiten eine Alternative zu den normalen Web-Frontends bieten können. Sämtliche Beispielanwendungen dieses Beitrags finden sich als Demoprogramme auf unserer Projektseite http://bsd119.ib.hu-berlin.de/~ft/.

Die wesentlichen Nachteile der APIs wie Performance und Limitierung kommen eigentlich erst bei größeren Untersuchungen zum Tragen. Besonders die Limitierung wird immer wieder als Grund dafür angeführt, warum die APIs für Untersuchungen nicht verwendet werden. Bezüglich der Zuverlässigkeit hat Google mit dem Wechsel zur AJAX API gezeigt, dass man sich nicht darauf verlassen sollte, dass die Dienste weiter (kostenfrei) erhalten bleiben.

Literaturangaben

[1] P. Bausch, Yahoo! Hacks, O'Reilly, 2005.
[2] T. Calishain, R. Dornfest, Google Hacks: 100 Industrial-Strength Tips and Tools, O'Reilly, 2003.
[3] P. Cimiano, A. Pivk, L. Schmidt-Thieme, S. Staab, Learning Taxonomic Relations from Heterogeneous Sources of Evidence. (2003). URL: http://www.uni-koblenz.de/~staab/Research/Publications/2005/OL-book-chapter-cimiano.pdf
[4] R.T. Fielding, Architectural Styles and the Design of Network-based Software Architectures. University of California Irvine. (2000). URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
[5] S. Harnad, T. Brody, F. Vallieres, L. Carr, S. Hitchcock, Y. Gingras, C. Oppenheim, H. Stamerjohanns, E. Hilf, The green and the gold roads to Open Access. (2004). URL: http://www.nature.com/nature/focus/accessdebate/21.html
[6] M. Henzinger, Search Technologies for the Internet, Science 317 (2007), 468-471.

[71] http://perlmeister.com/snapshots/200506/index.html
[72] http://bsd119.ib.hu-berlin.de/cgi-bin-gs/slideshow.cgi

[7] P. Mayr, Das Dateiformat PDF im Web - eine statistische Erhebung, Nachrichten für Dokumentation 53 (2002) No. 8, 475-481. URL: http://www.ib.hu-berlin.de/~mayr/arbeiten/IWP-2002.pdf

[8] P. Mayr, F. Tosques, Webometrische Analysen mit Hilfe der Google Web APIs. Information - Wissenschaft & Praxis 56 (2005a) No. 1, 41-48. URL: http://www.ib.hu-berlin.de/~mayr/arbeiten/Mayr_Tosques_IWP05.pdf

[9] P. Mayr, F. Tosques, Google Web APIs - An Instrument for Webometric Analyses? pp. 677-678. In: Ingwersen, Peter; Larsen, Birger (eds.): 10th International Conference of the International Society for Scientometrics and Informetrics. (2005b). Stockholm (Sweden) URL: http://www.ib.hu-berlin.de/~mayr/arbeiten/ISSI2005_Mayr_Toques.pdf

[10] P. Mayr, A.K. Walter, An exploratory study of Google Scholar, Online Information Review 31 (2007) No. 6, 814-830. URL: http://www.ib.hu-berlin.de/~mayr/arbeiten/OIR31-6.pdf

[11] F. McCown, M.L. Nelson, Search Engines and Their Public Interfaces: Which APIs are the Most Synchronized? In: 16th International World Wide Web Conference. (2007). Banff, Canada URL: http://www2007.org/posters/poster868.pdf

[12] U. Ogbuji, Using WSDL in SOAP applications. An Introduction to WSDL for SOAP Programmers. (2000). IBM. URL: http://www.ibm.com/developerworks/library/ws-soap/

[13] L. Stein, Creating a bioinformatics nation, Nature 417 (2002) No. 6888, 119-120. URL: http://www.nature.com/nature/journal/v417/n6885/full/417119a.html

[14] M. Thelwall, Extracting Accurate and Complete Results from Search Engines: Case Study Windows Live, JASIST 59 (2007) No. 1, 38-50.

[15] F. Tosques, P. Mayr, Web Services - Einsatzmöglichkeiten für das Information Retrieval im WWW. pp. 175-188. In: Ockenfeld, Marlies (ed.): 27. DGI-Online-Tagung. (2005). Frankfurt am Main: DGI. URL: http://www.ib.hu-berlin.de/~mayr/arbeiten/tosques_mayr_dgi05.pdf