LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

INSTITUT FÜR INFORMATIK
LEHR- UND FORSCHUNGSEINHEIT FÜR
PROGRAMMIER- UND MODELLIERUNGSSPRACHEN

Institut für Informatik

Web Information Systems

François Bry, Norbert Eisinger, Tim Furche

Lecture notes, November 4, 2009
http://www.pms.ifi.lmu.de/publikationen

François Bry, Norbert Eisinger, Tim Furche

Web Information Systems

Ludwig-Maximilians-Universität München, November 4, 2009

Contents

Part I

Unstructured Data or The Web of Documents: HTML

CHAPTER A.1

Basics: HTML Web and Information Retrieval (IR)

A.1.1 Representing textual data and documents: Markup

A.1.1.1 SPAGHETTI MODEL

Text formatting or text justification tools (in the 60es and 70 es of the 20th century) and early word processors (in the 70es and 80es of the 20th century) used data structures from programming languages, mostly arrays, for **representing textual documents**: Texts were internally stored as *lists of arrays*, an array representing a page, and an array element representing a character on a page. This was rather natural for, at that time, type writers with fixed width fonts, were omnipresent: With such type writers, texts were printed as bundles of pages, every page had (almost) the same number of lines, and every line (almost) the same number of characters.

This approach had serious *drawbacks*. First, data structures from programming languages, such as arrays, are language dependent. Second, text organized in pages and lines, requires complicated algorithms for common operations such as cut, past, changes of page size, and insertion of footnotes. Third, combining text portions from languages written in different directions (like English from left to right and Arabic from right to left, or like Chinese both vertically from right to left and horizontally from left to right) requires involved algorithms.

As a consequence, another model, sometimes called **"spaghetti model"**, has emerged (during the 70s of the 20th century) for the representation and storage of textual documents. With this model a text is represented as a **sequence of both ordinary characters**, that is, those of the text to represent, **and control characters** or control strings. Control characters or strings either express symbolic addresses (pointers) making cut and paste operations updates easy, or are used for structure specifications (for example a control character specifies the beginning of a paragraph, another its end), or are used for rendering specifications.

EXAMPLE: **CONTROL CHARACTERS**

To indicate that a portion of text is to be rendered bold, it is surrounded by specific control characters. Up to the control characters or strings used, this looks like the following example (in a SGML or XML syntax):

```
1    <paragraph>This approach had <bold>serious</bold>drawbacks.</paragraph>
```

The set of characters of the old ASCII character encoding contains, out of 127 characters, 32 control characters (with the decimal code 0–31). Most modern character encodings also contain these

characters to maintain compatibility with ASCII. An example of such a control character is the bell character (decimal code 7): It was originally developed as part of the Baudot codes to alert an operator on the other end of a telegraph line, e.g., to indicate the end of a message. ASCII line termination characters (*line feed* and *carriage return*) are used by most operating systems to indicate line termination in text files. For historical reasons (in typewriters *carriage return* returns the paper roll to the left margin and *line feed* advances the paper roll to the next line) Microsoft operating systems use both characters to indicate a line termination, Linux uses only *line feed*, Apple operating systems only *carriage return*.

Note that, with the spaghetti model, cut and paste operations are not as simple to implement as it seems. Indeed, they require garbage collection for freeing the memory from portions of text no longer pointed to by a document. However, garbage collecting is an issue well understood (cf. compilers).

A.1.1.2 MARKUP LANGUAGES

Structure and rendering in texts modeled after the spaghetti model are specified using control characters or strings inserted in the text. “Markup” (one word) denotes such control characters or strings. In contrast, “Mark up” (two words) denotes hand-written typo corrections like the following:

Markup languages have been defined for:

- formatting texts
- representing the logical (or semantical) structure of texts
- data interchange

One can distinguish between the following three types of Markup Languages:

- Specific or layout-oriented markup languages
- Generalized or structure-oriented markup languages
- Generic markup languages or meta markup languages

A.1.1.3 SPECIFIC OR LAYOUT-ORIENTED MARKUP LANGUAGES

Specific or layout-oriented markup languages (like PostScript from Adobe) usually have control characters, also called instructions, of three kinds:

- instructions changing the appearance of the character immediately preceding or following it.
- instructions changing the appearance of all characters up till an associated end instruction.
- instructions changing the appearance of the page (or similar logical entity such as paragraph, section) they occur in or following them.

Often, *well-formedness conditions* are not, or insufficiently, specified in specific or layout-oriented markup languages. A sequence such as

```
1     <italics>aaaa<bold>oooo</italics>cccc</bold>
```

might be allowed in specific or layout-oriented markup languages and might render the string `cccc` either in bold, or not in bold. (In contrast, SGML and XML, the syntax of which is used in the preced-

EXAMPLE: **“MARK UP” FOR TYPOGRAPHIC PROOFS**

Instruction to printer	Mark	Examples	
		In the text	*In the margin*
Character or word to be deleted		Council and Parliament	
Characters to be corrected		There are several errors to correct here.	
Group of characters to be corrected		Letters to be corrected.	
Character or word to be added		A word missing.	
Text to be added		1. January 12. December	
Transpose characters or words	or	These letters are transposed.	
Move words or groups of words		To move one or lines or paragraphs, this mark is used, more words.	
Transpose lines		Transposed. These lines are	
Add space between words	or	A space is missing here.	
Reduce space between words		These spaces are too big.	
Close up		A space is wrong here.	
Delete and close up		Council	or
Delete and keep space		An action plan	
Add space between lines		These lines are too close together	
Reduce space between lines		These lines are too far apart.	
Text to be raised or lowered		This line is very uneven.	
Text to be aligned (move to the right)		The first line of text is too far to the left.	
Text to be aligned (move to the left)		The second line of text starts too far to the right	
Text to be centred		This text should be centred	
Create new paragraph		... line. A new paragraph should begin here.	
Text to run on (no new paragraph)		... line. No new paragraph here.	
Take back to previous line		This hy- phen should be avoided	
Take forward to next line		This hyphen is bad- y placed.	
Change to italic		Ad infinitum	ital.
Change italic to roman		*Status quo*	rom.
Change capitals to lower case		UNESCO	l.c.
Change to capitals or small caps		Robert burns, AD 1759-96	caps s.c.
Change to bold face		This word is important	bold
Superior character required		The Court's judgment (1).	1
Inferior character required		CO_2	2
Stet (let original text stand)	.....	This correction was in error	

NB: — A correction made in the text must always have a corresponding mark in the margin; otherwise it may be overlooked when the corrections are made.
— Where instructions or comments are written in the margin, they must always be encircled to show that they are not to be printed.
— Where there are several mistakes in one word, it is better to rewrite the whole word, especially if the word has no more than three or four letters.

Source: [43]

Figure A.1.1: “Mark up” for typographic proofs

ing example, forbid such a sequence because it is not not well-formed, see below.) As a consequence, information is often lost in converting texts from one layout-oriented markup language into another.

A.1.1.4 GENERALIZED OR STRUCTURE-ORIENTED MARKUP LANGUAGES

The separation of document structure from layout characterizes **generalized or structure-oriented** markup languages. This separation has been first proposed in a research project at IBM at the end of the 60s of the 20th century with the language DCF GML (Document Composition Facility Generalized Markup Language):

C. F. Goldfarb, E. J. Mosher, and T. I. Peterson. An online system for integrated text processing. In *Proc. American Society for Information Science*, volume 7, pages 147–150, 1970 [25]. .

(With DCF GML were both the name "markup language" and language acronyms ending with "ML" used for the first time.)

The language **DCF GML** offers:

- Concepts such as "chapter" and "paragraph" for a specification of the logical structure of documents.
- Instructions for the specification of document parts to be automatically numbered (like chapters and list items), or to be automatically referred to (like literature references).
- A layout (or formatting) of document parts like "chapter" and "paragraph" with limited choices (e.g. choices of fonts and of number styles like Arabic or Roman).

It is worth mentioning that with DCF GML both, the document structure and the interpretation (or translation) of this structure in some layout were fixed and could not be modified.

Markup languages like RTF (Microsoft Word), LaTex and HTML combine aspects of specific and generalized markup languages. They have for example markup for structure (such as titles) as well as markup for layout (such as bold).

LOGICAL STRUCTURE VS. LAYOUT There are several advantages in keeping **distinct the specification of the logical structure** of a document from the specification of its layout:

- A same document might have *various distinct layouts*, for example for renderings on screens of different sizes: On large screens, more sophisticated renderings are possible than on small screens.
- Not all renderings are of the same nature. In addition to printing on screens or paper, there also are vocal and Braille (tactile) renderings for persons with disabilities.
- Keeping the logical structure and the layout specifications separate increases *data independence*: It eases modifying either of logical structure or layout, for example changing the layout for design reasons.

  For instance, HTML documents that use mainly elements for logical structuring rather than formatting elements are much easier to analyse automatically. Thus such documents are often easier to find with modern Web search engines. This point has, in fact, become the main driver for using the generalized features of HTML (cf. "search engine optimization").

Note that, in some cases, a specific rendering might require to change the logical structure of a document. This is for example the case of a tables used for time tables. A table with columns for days and rows for hours is convenient a rendering on a large screen. It is inconvenient because hardly readable on the small screens of PDAs or portable phones. For rendering on small screens, a sequence of hours of a day and of days is preferable it makes it easy to scroll (through the hours of a day and through the days).

A.1.1.5 GENERIC MARKUP LANGUAGES OR META MARKUP LANGUAGES

So far, SGML and XML are the only generic or meta markup languages.

Meta markup languages offer formalisms inspired from tree grammars, a special kind of context-free grammars, using which the structure of a text can be specified. In SGML for example, the structure of an address book can be specified by a so-called DTD, short for Data Type Definition, as follows:

```
<!ELEMENT address-book card* >
<!ELEMENT card (person, email+, comment?) >
<!ELEMENT person (title?, first-name, nickname?, last name) >
<!ELEMENT last-name (#PCDATA) >
<!ELEMENT first-name (#PCDATA) >
<!ELEMENT nickname (#PCDATA) >
<!ELEMENT email (#PCDATA) >
<!ELEMENT comment ANY >
```

The keyword ELEMENT at the beginning of a line expresses that the line specifies the structure of an SGML or XML element, the name of which is given after that keyword. The remaining of a line specifies the so-called "content model" of the element using regular expressions. `#PCDATA` stands for string content.[1] `ANY` means that any combination of (parsable) strings and/or elements is an allowed content model. (`ANY` is useful in designing complex DTD for first leaving unspecified some content models.)

In SGML and XML, **element names** (like "person") are used for forming so-called **opening and closing tags** (like `<person>` and `</person>` respectively.

It is worth stressing that `#PCDATA` is the only scalar data type—using a denomination of programming languages—for elements of SGML. For attributes, there are also the scalar data types CDATA, NMTOKEN (name token, i.e., an identifier), ENTITY, NOTATION, ID, and IDREF.

Markup languages specified by DTD or similar formalisms are called **SGML or XML applications**.

> EXAMPLE: **XML APPLICATION**
>
> XHTML 1.0, for example, the re-definition of HTML 4.01 in XML is an XML application.

SGML is a standard (ISO 8879:1986 SGML) of the ISO (International Organization for Standardization). XML is a recommendation[2] of the World Wide Web Consortium W3C:

> Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (Fifth Edition). Recommendation, W3C, 2008. http://www.w3.org/TR/REC-xml/, checked 2009/10/14 [6]. .

Historically, SGML goes back to DCG GML as follows: The ANSI (American National Standard Institute) first specified a generic markup language inspired from DCF GML. Later, this generic markup language was refined and resulted in SGML. The following remark on the origins of SGML is worth citing:

[1] The name means *parsed* character data to indicate that the content is tokenized and parsed to recognize possible nested elements or to resolve entities like `&lt;` (the < sign) or `&ldquo;` (the left double quote " in HTML). In contrast, so called `CDATA` sections contain unparsed character data and thus may directly contain reserved characters like < or &. In other words, in `#PCDATA` all reserved characters are unquoted and thus are interpreted by a parser, in `CDATA` sections all characters are already quoted.

[2] The W3C calls its standards "recommendations" for stressing a change it introduced in standardization of computer formalisms and languages. Before the W3C, standardizing used to be initiated after a formalism or language was established and to be a slow process. The W3C strives for a priori standards, which is conceived by a joint effort of the main actors.

> "It appears certain to me that at least these three ideas were common already in the 1960's, often within distinct communities which rarely talked to each other: (a) the notion of separating 'content and structure' encoding from specifications for [print] processing; (b) the notion of using names for markup elements which identified text objects 'descriptively' or 'generically'; (c) the notion of using a (formal) grammar to model structural relationships between encoded text objects." Robin Cover, SGML: General Introductions and Overviews, http://xml.coverpages.org/general.html#faq (accessed 9 April 2009)

TAG OMISSION. **Tag omission** for document minimization is a feature of SGML which has not survived in XML.

Tag omission Consider the following excerpt of an SGML document:

```
<persons>
  <person>
    <title>Prof. Dr.</title>
    <first-name>FranÃğois</first-name>
    <last-name>Bry</last-name>
  </person>
  <person>
    <title>Dr.</title>
    <first-name>Norbert</first-name>
    <last-name>Eisinger</last-name>
  </person>
  <person>
    <title>Dr.</title>
    <first-name>Tim</first-name>
    <last-name>Furche</last-name>
  </person>
</person
```

SGML's tag omission would make it possible to express this excerpt as follows:

```
<persons>
  <person>
    <title>Prof. Dr.</title>
    <first-name>FranÃğois</first-name>
    <last-name>Bry</last-name>
  </person>
  <person>
    Dr.</title>
    <first-name>Norbert</>
    <last-name>Eisinger
  </person>
  <>
    Dr.</title>
    Tim</first-name>
    <last-name/Furche/
</persons>
```

Tag omission reduces the length of, or "minimizes", an SGML document. However, it makes parsing significantly more complex. Indeed those omissions of tags that are allowed, that is those that are

non-ambiguous, cannot be expressed with a context-free grammar. Furthermore, memory space being now widely available on a much larger scale than in the SGML age, document minimization is no longer a stringent need. For these two reasons, tag omission is not allowed by XML.

DIFFERENCES BETWEEN XML AND SGML. If we use the term "XML" properly, it denotes the core XML specification [6]. XML proper is a *significant* simplification of SGML, but does not introduce any significant new features. However this simplification proved to be so significant that quickly an ecosystem of other technologies has developed (following `http://www.w3.org/XML/Core/`): XML Namespaces [5], XML Infoset [12], XInclude [36], XLink [14], xml:id [37], "Assigning Style Sheets with XML Documents" [11], and XML Base [35]. Sometimes we broaden that even further to also include XML Schema [19, 48, 3] and XLink [14]. In the following, we often refer to the entire ecosystem also simply as "XML".

If we compare SGML with not only XML proper but the whole set of technologies, a number of new features stand out:

- XML has hypertext features while SGML has none.
- XML has a more sophisticated type definition language, called XML Schema, than the DTD language of SGML.
- XML has a rich collection of scalar types (called simple types) while SGML only has a single scalar type (string called `#PCDATA`) for elements.

OUTLOOK: **DATA SERIALIZATION LANGUAGES**

There is also a class of *generic* markup languages tailored towards **data (structure) serialization** rather than document markup. Languages in this class, such as ASN.1 (*Abstract Syntax Notation One*) and YAML (*YAML ain't markup language*), are also generic markup languages, but do not allow "mixed content" that is markup where strings and elements (i.e., structured data) are mixed as content of another element:

```
<p>This is <em>mixed</em> content.</p>
```

In languages such as YAML, the content of an element must be either all string or all element.

The reason for this limitation is that these languages are primarily designed to describe and exchange *programming language data structures* such as records (or C `structs`), lists and arrays.

ASN.1 [28] is a standard by the ITU-T (and the ISO/IEC) that is used in telecommunication and networking to describe the type of data that is exchanged in a network protocol.

The following example shows an ASN.1 schema (or structure definition). ArilineFlight is defined as a sequence of records. Each record records the name of the airline, the flight number, (nested) sequences describing the seats in the flight and the airports the flight visits, the crewsize (either 6, 8, or 10), and a flag cancel indicating whether the flight has been canceled.

```
AirlineFlight  ::=  SEQUENCE
      {
       airline   IA5String,
       flight    NumericString,
       seats     SEQUENCE
                     {
                      maximum   INTEGER,
                      occupied  INTEGER,
                      vacant    INTEGER
                     },
```

```
11      airport    SEQUENCE
                   {
13                  origin              IA5String,
                    stop1       [0]     IA5String  OPTIONAL,
15                  stop2       [1]     IA5String  OPTIONAL,
                    destination         IA5String
17                 },
        crewsize ENUMERATED
19                 {
                     six    (6),
21                   eight  (8),
                     ten    (10)
23                 },
        cancel   BOOLEAN    DEFAULT FALSE
25      }.
```

How data about a specific flight (following the above schema) is represented, depends in ASN.1 on the *encoding rules*. Most encodings represent ASN.1 data as binary data. For example, in DER (*distinguished encoding rules*) data is encoded as triples of binary codes, the first component indicating the type of data, the second the length, and the third the actual value. For more details see [32] and the examples in [9].

YAML [2] is a more recent example of a data serialization language used, e.g., in many Ruby frameworks such as Rails. Its sublanguage JSON [13] has become the second most used format for data transmission in Web 2.0 applications after XML.

The disadvantage of data serialization languages is that mixed content as above must be represented with additional tags:

```
1     <p><normal>This is </normal><em>mixed</em><normal>content.</normal></p>
```

This is represented in YAML as:

```
1 p:
    - normal : "This is "
3   - em:      "mixed"
    - normal:  " content."
```

This difficulty with representing mixed content is one reason why HTML values are represented as unstructured, opaque strings in YAML and JSON. Though this allows easy inclusion of arbitrary HTML it makes it hard to process (e.g., reformat) the HTML contained in YAML or JSON data.

REFERENCES

ITU-T. Abstract syntax notation one (asn.1): Specification of basic notation. Recommendation X.680, ITU-T, 2008. http://www.itu.int/rec/T-REC-X.680/en, checked 2009/10/14 [28]. .

John Larmouth. *ASN.1 Complete*. Morgan Kaufmann, 1st edition, 1999. http://www.oss.com/asn1/dubuisson.html, checked 2009/10/14 [32]. .

Lillian N. Cassel and Richard H. Austing. ASN.1. Tutorial webpage, 2000. http://www.obj-sys.com/asn1tutorial/asn1only.html, checked 2009/10/14 [9]. .

Oren Ben-Kiki, Clark Evans, and Ingy döt Net. Yaml ain't markup language (yaml) version 1.2. Informal specification, 2009. http://www.yaml.org/spec/1.2/spec.html, checked 2009/10/14 [2]. .

D. Crockford. The application/json media type for javascript object notation (json). RFC (Informational memo) RFC 4627, IETF, 2006. http://www.ietf.org/rfc/rfc4627.txt, checked 2009/10/14 [13]. .

A.1.2 HTML and XHTML

WHAT IS HTML? The W3C XHTML2 Working Group home page (http://www.w3.org/MarkUp/) describes HTML as follows:

> "HTML is the lingua franca for publishing hypertext on the World Wide Web. It is a non-proprietary format based upon SGML, and can be created and processed by a wide range of tools, from simple plain text editors - you type it in from scratch - to sophisticated WYSIWYG authoring tools."

WHAT IS XHTML? The W3C XHTML2 Working Group home page gives the following answer:

> "The Extensible HyperText Markup Language (XHTML) is a family of current and future document types and modules that reproduce, subset, and extend HTML, reformulated in XML rather than SGML.
>
> XHTML 1.0 was the W3C's first Recommendation for XHTML, following on from earlier work on HTML 4.01, HTML 4.0, HTML 3.2 and HTML 2.0. With a wealth of features, XHTML 1.0 is a reformulation of HTML 4.01 in XML, and combines the strength of HTML 4 with the power of XML."

Currently, a new version of HTML (and XHTML) is in preparation, HTML 5. A first working draft of HTML 5 has been published in January 2008.

The following focuses mostly on XHTML 1, the newest standardized version of HTML so far.

A.1.2.1 HISTORY OF HTML

1989: Tim Berners-Lee and Robert Caillau at CERN (Geneva) started developing an information system linking documents in TEXand PostScript. Tim Berners-Lee developed HTML as a very simple language inspired from SGML.

1991: `www-talk` mailing list launched by CERN, document titled "HTML Tags" published. It listed 20 tags, or element names, 13 of which remain in HTML 4, and the structure of an HTML document. HTML was seen as an SGML application: A DTD was specified for HTML. SGML's tag omission (and a lack of experience) led to a clumsy language:

```
    <br> instead of XHTML's <br/>
2   <p>  instead of XHTML's <p> ... </p>
    <li> instead of XHTML's <li> ... </li>
```

that is, no closing tags `br` (line breaks), `p` (paragraph) and `li` (list item) elements.

Marc Andreessen and Eric Bina at National Center for Supercomputer Applications (NCSA, Urbana-Champaign) develop the X Window Mosaic browser.

(The original) HTML has never been standardized even though a standard proposal had been submitted.

1993-1994: HTML+: Extension of HTML *with tables* by Dave Raggets (1993: internal draft; 1994 publication at the WWW'1 Web Conference) Like HTML, HTML+ has never been standardized even though a standard proposal had been submitted.

Oct. 1994: Tim Berners-Lee founds the *World Wide Web Consortium* (W3C) at MIT (Massachusetts Institute of Technology) in collaboration with CERN with support from DARPA and the European Commission.

(INRIA (*Institut National de Recherche en Informatique et Automatique*) became "European W3C host" in 1995, the Keio University (Tokyo and Shonan Fujisawa Campus near Yokohama) the "Asia W3C host" in 1996. ERCIM (European Research Consortium in Informatics and Mathematics) took over the role of European W3C host in 2003.)

1995: HTML 2.0 specification RFC 1866 by the HTML Working Group of the IETF (Internet Engineering Task Force) on a proposal by Tim Berners-Lee. HTML 2.0 is not fully backwards compatible with former HTML versions.[3]

Jan. 1997: HTML 3.2 First W3C Recommendation for HTML

Dec. 1997: HTML 4.0 W3C Recommendation (slight editorial changes in April 1998) HTML 4.0 offers three flavors:

- Strict, in which deprecated elements are forbidden,
- Transitional, in which deprecated elements are allowed,
- Frameset, in which mostly only frame related elements are allowed.

Dec. 1999: HTML 4.01 W3C Recommendation a (limited) correction of HTML 4.0

2000: ISO HTML (ISO/IEC 15445:2000) based on HTML 4.01 Strict) ISO/IEC 15445:2000 is a subset of HTML 4, standardized by ISO/IEC. It is stricter than HTML 4 by requiring for example an `h2` element to exists between an `h1` element and an `h3` element. Though never adopted by browsers, it plays some role for long-term archival of HTML documents.

XHTML a *reformulation of HTML 4.01 in XML 1.0*, revised 2002

Jan. 2008: HTML5 published by the W3C as a working draft. HTML5 significantly modifies and extend former versions of HTML as follows:

- there are tags for *video and audio files* in common formats and for inputs of various kinds (dates, phone numbers, etc.)
- there is an *event model* coupled to JavaScript APIs for dynamics (like drag and drop or copy and paste) aiming at making obsolete plug-in-based so called "rich Internet application (RIA)" technologies (based for example on Adobe Flash, Microsoft Silverlight, or Sun JavaFX)
- the canvas element make it possible to specify a surface for graphics programmed in JavaScript

Furthermore, HTML5 further differs from former versions of HTML as follows:

- the semantic tags (like `div`, `span`, `section`) are aligned with today's practice;
- HTML5 is no SGML application (even though it keeps a syntax reminding of SGML);
- HTML5 has both an HTML syntax denoted HTML5 and an XML syntax denoted XHTML5;
- In its HTML syntax, HTML5 properly specifies the character encoding ((1) at the transport level using for example the HTTP Content-Type header, or (2) using a Unicode Byte Order Mark (BOM) character at the start of the file, or (3) using a meta element with a charset attribute);

[3] HTML 1 or HTML 1.0 never existed.

- The HTML syntax of HTML5 allows for MathML and SVG elements;
- purely layout elements and attributes of former versions of HTML no longer exist (or have been redefined) in HTML5.

THe HTML dialect of HTML5 (but not XHTML5) differs from XHTML in allowing some limited tag omission (like `<br>` instead of `<br/>` and `<li>` instead of `<li> ... </li>`), the goal being of making HTML5 backwards compatible. (This can be interpreted as a recognition of a failure of XHTML.)

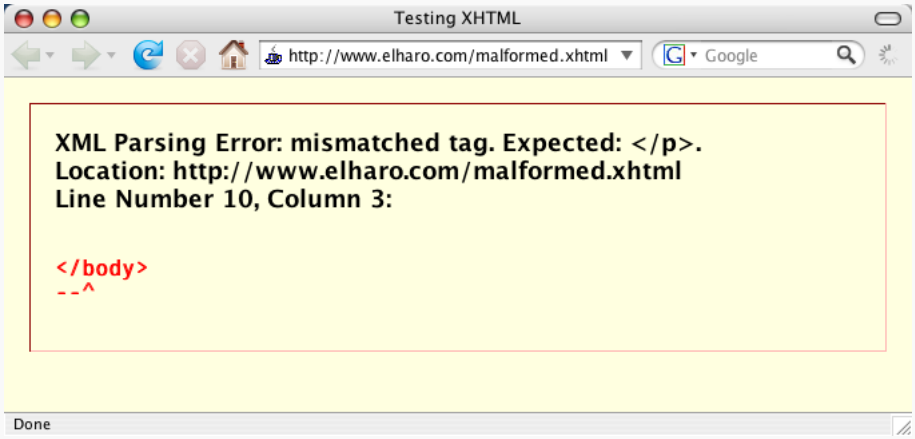
HTML is actively supported by Google and Apple (the editors of the HTML 5 draft recommendation, Ian Hickson and David Hyatt, are with Google and Apple respectively) and many other companies. Microsoft has only recently started to contribute.

HTML5 is not expected to become a recommendation soon, keeping with the approach "first implement, then standardized".

OUTLOOK: WHY HAS XHTML 1 NOT REALLY CAUGHT ON?

One of the major reasons, why XHTML 1 has not caught on (in contrast to the expectation of the W3C and many XML experts), is the *handling of markup errors*, for instance, when elements are not properly nested, attributes not quoted, ...

XHTML 1 is a proper XML application and, as such, uses the same parsing rules as any XML application. Part of these rules is that certain errors *are required* to raise an error. The following is an example of an XHTML 1 document with an error in its markup (as rendered by Firefox):



Though strict error handling can be a virtue (as it helps overcome the attitude that if a document looks fine in a browser it is fine for publishing), in the case of XHTML 1 it has been damning:

- If we were to apply this error handling to *existing documents*, many (in fact: most) documents would not display. Therefore, browser developers are forced to provide two modes ("quirks mode" and "strict mode") and heuristics which document to render in which mode.
- Even for *new documents*, neither users not even application developers seem to be willing to be bothered to provide "perfect" XHTML. This may be due to the fact that browsers have been doing a fairly good (though very intricate) error recovery.

REFERENCES:

Steven Pemberton and the W3C XHTML2 Working Group. Some early ideas for HTML. Information web page, 2009. http://www.w3.org/MarkUp/historical, checked 2009/09/16 [41]. .

Wikipedia. History of HTML. Wikipedia page, 2009. http://en.wikipedia.org/wiki/HTML#History_of_HTML, checked 2009/09/16 [53]. .

Sam Ruby, Chris Wilson, Michael Smith, and Dan Connolly. W3C HTML working group. Home page, 2009. http://www.w3.org/html/wg, checked 2009/09/16 [45]. .

Ian Hickson and David Hyatt. HTML 5: A vocabulary and associated APIs for HTML and XHTML. Working draft, W3C, August 2009 [27]. .

Anne van Kesteren. HTML 5 differences from HTML 4. Working draft, W3C, August 2009 [51]. .

A.1.2.2 STRUCTURE OF AN XHTML DOCUMENT

An XHTML document is structured as follows:

CONCEPTUALLY:

1. XML declaration
2. Prolog (specifies the DTD together with the name of the document element, here `html`)
3. `html` element (also called "document element")
4. `head` element (ordering of child elements irrelevant)
   - `base` element (base pathnames for all relative URIs in the document)
   - `meta` element(s)
   - `title` element (the content of which is displayed in browser windows)
   - `link` element(s) (specifies a relationship to another document)
   - `script` element(s) (for embedding or referencing a script (= Javascript program) in the document)
   - `style` element (for embedding or referencing a style sheet in the HTML document)
5. `body` element (the text proper)

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"></meta>
    <meta name="Author" content="Fran&ccedil;ois Bry" />
    <title>
      History of HTML
    </title>
    <link rel="stylesheet" type="text/css" href="/pmsmail.css"></link>
    <script src="/pmsmail.js" type="text/javascript"></script>
    <style type="text/css">
```

```
        HIER CSS RULES
17    </style>
    </head>
19  <body>
        HTML's first version was conceived ...
21  </body>

23 </html>
```

The prolog of the XHTML document given above can be explained as follows:

- `Strict`: Indicates only XHTML without (deprecated) tags from former versions, no frames.

  Alternatives to STRICT are as follows:

```
1     "-//W3C//DTD HTML 4.01//EN"                HTML 4
      "-//W3C//DTD HTML 4.01 Transitional//EN"   HTML 4 and deprecated tags
3     "-//W3C//DTD HTML 4.01 Frameset//EN"       frameset and frames
      "-//W3C//DTD XHTML 1.0 Strict//EN"          XHTML
5     "-//W3C//DTD XHTML 1.0 Transitional//EN"    XHTML and deprecated tags
      "-//W3C//DTD XHTML 1.0 Frameset//EN"        frameset and frames
```

- `EN`: Though this indicates the language of DTD, this is the *only normative* version, i.e., there is not German or Latin HTML *DTD*. Note, that the language of the *document* can be different from en (specified, e.g., with `xml:lang='en'` attribute of the `html` element.

HEADINGS: ELEMENT DELIMITERS

Headings (that is, section titles or sub-titles): `h1` to `h7` elements. Note that XHTML has a *title* element (in the head). Section titles or sub-titles are called in HTML "headings".

Warning: HTML headings *precede* the portion of text they are the title of. An alternative structure would consist in having a title element together with and before a text element within a section-like element. Such a structure is not only be more natural for a human reader but also makes querying and processing documents easier. (XHTML has a `div` (for division) element, HTML5 a `section` element, making it possible—but not mandatory—to structure an HTML5 document in this alternative manner. Since such a structuring is not mandatory, the burden remains on XHTML and HTML5 querying and processing)

XHTML titles are as follows:

```
  <h1>History of HTML</h1>
2   <h2>The beginning</h2>
      <p> ... </p>
4     <p> ... </p>
    <h2>The first HTML standards</h2>
```

Preferable, because more logical and easier to process, would be (similar to DocBook, an SGML application for technical documents):

```
1     <div>
        <h1>History of HTML</h1>
```

```
        <div>
           <ch2>The beginning</h2>
           <p> ... </p>
           <p> ... </p>
        </div>
        <div>
           <h2>The first HTML standards</h2>
           ...
        </div>

        ....

    </div>
```

TREE STRUCTURE OF AN XHTML DOCUMENT:

- **Nodes** correspond to both elements and attributes.
- There is one single root. Note that this root node is above of the (element) node for the `html` element.
- id-refs and links are not considered in the tree structure of an XHTML document.

See XML Infoset [12] for more details.

A.1.2.3 SELECTED XHTML ELEMENTS AND ATTRIBUTES

Elements for "divisions", or document sections, and elements for "span", that is, inline portions of text also called text fragment.

A selection of HTML elements (briefly named, explained where necessary):

Paragraph: non-recursive (no nesting of one `p` in another `p`)

```
    <p> ... </p>
    <p  dir="ltr"> ... </p>                                    direction left-to-right
```

Division: of a document, possibly recursive.

```
        <div> ... </div>
```

Span: of inline characters, possibly recursive

```
        <span> ... </span>
```

Class attributes: mainly for use with CSS.

```
        <div class="explanation"> ... </div>
```

With a CSS rule for a specific rendering div.explanation background-color: grey;

```
<span class="warning"> ... </span> possibly with CSS rule
```

span.warning color: red;

Emphasize: possibly recursive, if not otherwise specified (in a style sheet) rendered in *italics* by most browsers

```
<em> ... </em>
```

Other inline elements: `strong`, `cite`, `code`, `var`, ...

Image:

```
<img src="..." alt="Alternative text" ...> ... </img>
```

Line break:

```
<br/>
```

Horizontal rules:

```
<hr/>
```

Ordered and unordered lists:

```
<ol>                    or          <ul> ...
   <li> ... </li>
   <li> ... </li>
</ol>
```

Anchor (of hypertext links): Simple hypertext link contained in source page (anchor in source page). `href` attributes gives goal page (and possibly a position in goal page using a fragment indicator `#history`). source page anchor in source page goal page position in goal page

```
<a href="..."> ... </a>
```

Fragment indicators are mapped to named anchors (`a` with `name` attribute) or `id` attributes of XHTML elements:

```
<h2 id="section2">Section Two</h2>
...later in the document
<p>Please refer to <a href="#section2"> ... </p>
```

or

```
<h2 id="section2"><a name="explanation">Section Two</a></h2>
...later in the document
<p>Please refer to <a href="#section2"> ... </p>
```

- If both `id` and `name` are used with an a element, then they must have the same value.
- `name` can only be used on a elements, `id` on (nearly) all elements.
- Some (very, very) old browsers do not support the `id` attribute
- `name` values are less restricted than `id` values (in particular, character references like `&ccedil;` are allowed in `name` values, not in `id` values, though this practice is discouraged even in HTML 4).

OTHER XHTML FEATURES:

- **Entities** (such as `&ccedil;`) for specifying special characters.
- **Forms** for entering data.
- **Tables** (note that they are in HTML line-wise, so as to make it possible to display the top of not yet fully downloaded table.).
- **Frames** for presenting documents in "multiple views" so as to make it possible to change the content of the one and keeping the content of others.
- Many so-called “deprecated” elements and attributes (from former HTML versions for expressing layout properties like sizes, colors, italics, etc.) kept for backwards compatibility reasons.

REFERENCES:

Steven Pemberton and the W3C HTML Working Group. Xhtml 1.0: The extensible hypertext markup language. Recommendation, W3C, 2000 [40]. .

Dave Raggett, Arnaud Le Hors, and Ian Jacobs. Html 4.01 specification. Recommendation, W3C, 1999 [44]. .

A.1.3 Hypertext models

Because the Web is widespread and HTML is the lingua franca of the Web, one may think that there are no other "hypertext model" than that of HTML. This is not the case, even though the success of HTML does not leave much room for other hypertext models than HTML's one.

HTML's hypertext model has some drawbacks, most notably:

- it is limited to a simple kind of hypertext links
- HTML's link specifications are located with their anchor (in the source document)

More complex links might be useful for expressing n-ary relations like for example pointing to explanations in different languages.

HTML links are difficult both, to *navigate backwards* and to *manage*. Backward navigation beyond that made possible by a history stack is useful in practice. If a link for example points to an explanation, it might be useful to following it backwards for finding what it is explaining. If a Web site relate each occurrence of a person name to this person's page, then it would be preferable, because much easier to update, to have a single occurrence of a specification of links from the person's name to the person's page. Such specifications are often kept together in so-called "link bases".

In spite of its limitations and deficiencies, the hypertext model of HTML can now be seen as a "general agreement".

Other hypertext models offer:

- links with several targets
- link bases
- links that can be traversed backwards

Most notable **examples of hypertext models** different from HTML's one are as follows:

- **HyTime**, the hypermedia/Time-based Structuring Language, an SGML application.

  HyTime is a standard of the ISO/IEC. The first edition was published in 1992, the second in 1997 Some of the concepts formalized in HyTime were later incorporated into HTML and XML.

  Main traits of HyTime:

  - HyTime is an application of SGML: it defines element types that can be used with an SGML document types to provide Hypertext facilities.
  - HyTime's model is *independent of the rendering*
  - HyTime offers complex locating of document objects, possibly intentionally (that is, by queries), possibly extensionally (by locations). HyTime has rich means to express *locations in documents* (by means of so-called "coordinates").
  - HyTime offers *relationships*, that is hyperlinks, between document objects and *roles* these objects play in the relationship, for example "defined-as".
  - HyTime temporal properties for the scheduling of document objects

    C. F. Goldfarb. Hytime: A standard for structured hypermedia interchange. *Computer*, 24(8):81–84, 1991 [24]. .

- **XLink**, W3C recommendation for *links in XML documents*

- XLink has simple links (or “outbound”, that is Ãă la HTML) and *extended links*. XLink extended links can include inbound and/or third-party “arcs” (an XLink extended link is a collection of arcs. An XLink arc corresponds to an HTML links)
- XLink offers links that have *arbitrary* numbers of participating resources
- XLink offers *link bases* allowing for a better modeling and a better maintenance of hypertext document collections.

XLink has not been a success. This might comes from the fact that XLink's syntax and denominations are a bit unnatural and complicated. However, XLink is interesting to study and some of XLink's features might come back through RDF.

OUTLOOK: **TOPIC MAPS**

`@TODO: missing`, see [42].

Steve DeRose, Eve Maier, and David Orchard. XML Linking Language (XLink) Version 1.0. Recommendation, W3C, 2001 [14]. .

- *Further hypertext models:*

Steven J. DeRose and David G. Durand. The TEI hypertext guidelines. *Computers and the Humanities*, 29(3):181–190, May 2005 [15]. .

Frank Halasz and Mayer Schwartz. The dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994 [26]. .

Steven J. DeRose and Andries van Dam. Document structure and markup in the FRESS hypertext system. *Markup Lang.*, 1(1):7–32, 1999. http://www.stg.brown.edu/~sjd/fress.html [16]. .

Research on new, richer, more sophisticated Hypertext model *for the Web* is no longer lively since HTML is ubiquitous on the Web. But hypertext systems of *limited scopes*, like learning system, often benefit from more sophisticated Hypertext model than that of HTML. Furthermore, many features of sophisticated hypertext models are not, or *not easily adaptable* to the open context of the Web.

Two technologies in the area of hypertext models are nonetheless worth mentioning since they may impact on future versions of HTML:

- **Redirection technologies** (like Bit.ly and tinyUrl) needed for web applications like Twitter.
- **Typing of links** through the `rel` attribute of HTML anchor (i.e., `a`) elements.

  Some values for this attribute are given a common meaning in HTML 4, e.g., `rel="Next"` `rel="Prev"` `rel="Contents"` for describing relationships between a collection of pages that together make up, e.g., an article.

  In HTML 5 a number of additional such meanings is standardized, while the community continues to develop new uses for `rel` (e.g., to encode RDF triple in HTML, see [1]).

EXAMPLE: **REL-VALUES USED BY SEARCH ENGINES**

The following two values for `rel` are nowadays supported by most search engines:

- `rel="nofollow"` indicates that bots should not follow the link.
- `rel="canonical"` specifies which one is primary or “canonical”, in case the same content is accessible through different URIs.

A.1.4 CSS: Cascading Style Sheets

Simplicity and Web platform versatility are two remarkably well achieved objectives of CSS.

> "Cascading Style Sheets (CSS) is a simple mechanism for adding **style** (e.g. fonts, colors, spacing) to Web documents." http://www.w3.org/Style/CSS/
>
> "CSS has various levels and profiles. Roughly speaking, desktop browsers implement level 1, 2 or 3, other programs implement the appropriate profile for their platform: cell phone, PDA, television, printer, speech synthesizer, etc." http://www.w3.org/Style/CSS/

A higher CSS level, as a CSS version is called, extends lower CSS levels ensuring upward compatibility (where possible). Furthermore, all levels of CSS use the same core syntax. This allows applications to parse style sheets after CSS levels that did not exist at the time the application was developed (of course, the features added by the higher CSS level cannot be used by the application).

A.1.4.1 HISTORY OF CSS

October 1994: Håkon Wium Lie publishes the *first draft*, or proposal, of Cascading HTML Style Sheets. There were several other styling languages already developed, or under development, like DSSSL for SGML documents at ISO. Novel with CSS was to combine, or "cascade", rendering specifications of both, the author and of the viewer, a key requirement on the Web.

Another major novelty of CSS was a *strict simplification* of previous styling languages: CSS allows only to adorn nodes in an HTML (or XML) tree with *properties* such as color, font family etc. It does *not*, however, *allow transformations* of that tree.[4]

December 1996: Publication of CSS level 1, or CSS 1, a W3C Recommendation mostly for screen rendering, consists of about 50 properties.

February 1997: The "Cascading Style Sheets and Formatting Properties Working Group" is established at W3C

May 1998: Publication of CSS level 2, or CSS 2, a W3C Recommendation. A subset of CSS 2 are "color specifications tailored to the needs and constraints of TV devices". CSS 2 adds about 70 properties (among others for aural rendering and page brakes) to the about 50 properties of CSS 1.

May 2003: Publication of CSS TV Profile 1.0, a W3C Candidate Recommendation

October 2006: Publication of CSS Print Profile, a W3C Working Draft, A subset of CSS 2,1 "for for printing to low-cost devices".

December 2008: Publication of CSS Mobile Profile 2.0, a W3C Candidate Recommendation. A subset of CSS 2.1 "baseline for interoperability between implementations of CSS on constrained devices (e.g. mobile phones)".

September 2009: Publication of CSS level 2 Revision 1, or CSS 2.1, a W3C Recommendation.

[4] With one exception: The property `content` allows adding plain text before and after elements (using the `::before` and `::after` pseudo-elements).

Since 1999: CSS level 3, or CSS 3, is under development. CSS 3 is to include:

- revised and extended selector syntax
- multi-column layout
- "marquee" effect: the content of too large an element for a screen is animated and moves automatically back and forth
- animation

The following document is good a reference "current CSS" which combines CSS 2.1 with stable parts of the upcoming CSS 3:

> Elika J. Etemad. Cascading style sheets (css) snapshot 2007. Working draft, W3C, May 2008. http://www.w3.org/TR/css-beijing/, checked 2009/09/16 [18]. .

Indeed, many current browsers already implement a significant portion of CSS3. The above-mentioned CSS Snapshots presents itself as follows:

> "This document collects together into one definition all the specs that together form the current state of Cascading Style Sheets (CSS)." [18]

According to the above-mentioned CSS Snapshot 2007, a "valid CSS document" is one that conforms to the following specifications:

1. CSS Level 2 Revision 1 (including errata)
2. CSS Namespaces
3. Selectors Level 3
4. CSS Color Level 3

In the following, "CSS" is used in this sense, that is, it refers to the four specifications above. CSS can be used with HTML and with any structured document format, among other XML.

EXAMPLE: **FLAVOR OF CSS**

A CSS style sheet is an (ordered) sequence of rules. If an HTML document is to be styled, the following rules could be used:

```
body { color: black; background: white; }
h1 { font-size: 170%;  }
h2 { font-size: 140%; }
h3 { font-size: 120% ; }
h4 { font-weight: bold; }
h5 { font-style: italic; }
p { text-align: justify; }
p.citation { margin-left: 5%; margin-right: 5%; width: 80%; }
    /* for <p class="citation"> ... </p> */
```

Since HTML element names are not case sensitives, selectors are not case sensitive in CSS for HTML (and only for HTML). Thus, the first CSS rule above could be written for example as follows:

```
BODY { color: black; background: white; }
```

A **CSS rule** consists of two parts:

1. **A Selector:** the part before the left curly brace. It specifies the elements affected by the declaration.
2. **Declarations:** the ; separated list within the curly braces. They specifies the style by means of property/value pairs.

   A property/value pair has two parts separated by a colon:

   a) **Property:** the part before the colon,

   b) **Value:** the part after the colon.

The *color* specified by the rule

```
body { color: black; background: white }
```

is inherited by the descendant elements of the `body` element, except for those that redefine, or override, the values of the property color. The background is *not* inherited automatically. Whether a CSS property is inherited in such a manner is an important part of the definition of the CSS property. It is possible to force inheritance of properties such as background or border that are not inherited by default:

```
p { background: inherit; }
```

A.1.4.2 ASSIGNING A CSS STYLE SHEET

. . . TO AN HTML DOCUMENT

There are four ways to associate CSS styles with an HTML element or entire document:

1. **Inline style:** Apply a CSS style sheet to an *individual element* using the element's `style` attribute.

   ```
   <p class="warning" style="color: red; background: yellow">
     No lecture on 2 May 2008
   </p>
   ```

   Inline style is considered poor authoring practice.

2. **Internal style sheet:** *Document-wide* CSS style sheet, embedded into a document using the document's `style` element:

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
     <head>
       <meta http-equiv="Content-Type"
             content="text/html; charset=iso-8859-1"></meta>
       <meta name="Author" content="Fran&ccedil;ois Bry" />
       <title>
          History of HTML
       </title>
       <style type="text/css">
          h1, h2, h3 { font-style: bold }
          h1 { color: blue; text-align: center; }
          p { text-align: justify; }
       </style>
     </head>
   ```

```
  <body>
     ...
  </body>
</html>
```

The style element can be placed anywhere in the head of an HTML document.[5]

The attribute type of the `style` element specifies that a CSS style sheet is used (`<style type="text/css">...</style>`).

For old browsers (like Netscape's Navigator 1 and Microsoft Internet Explorer 2), that do not support CSS and the `style` element and display its content, it has become common practice to escape the content of `style` elements using comments. However, this is no longer recommended for XHTML as XML processors may silently remove comments.

```
    <style type="text/css">
       <!--
          h1, h2, h3 { font-style: bold; }
          h1 { color: blue; text-align: center; }
          p { text-align: justify; }
       -->
    </style>
```

3. **Imported style sheet:** *Document-wide, external* CSS style sheet contained in a separate resource that is imported using the CSS `@import` notation.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type"
        content="text/html; charset=iso-8859-1"></meta>
  <meta name="Author" content="Fran&ccedil;ois Bry" />
  <title>
     History of HTML
  </title>
  <style type="text/css">
     @import url(basicstyle.css)
     ...
  </style>
</head>
<body>
   ...
</body>
</html>
```

One or more `@import` directive can be placed at the beginning (and only at the beginning) of any CSS style sheet, in particular at the beginning of an internal CSS style sheet like in the example above. An import directive is interpreted by *literal replacement* with the given resource.

4. **Linked style sheet:** *Document-wide, external* CSS style sheet contained in a separate resource that is linked to using the HTML `link` element.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <meta http-equiv="Content-Type"
        content="text/html; charset=iso-8859-1"></meta>
  <meta name="Author" content="Fran&ccedil;ois Bry" />
```

[5] Before or after the `title` element which is not part of the document tree and thus not affected by CSS styles.

```
    <title>
      History of HTML
    </title>
    <link rel="stylesheet" type="text/css" href="basicstyle.css" media="all"/>
    <link rel="stylesheet" type="text/css" href="webstyle.css" media="screen"/>
    <link rel="stylesheet" type="text/css" href="paperstyle.css" media="print"/>
  </head>
  <body>
    ...
  </body>
  </html>
```

In this example, a basic style sheet holds for all media, additional ones extending the basic one for browsers and for paper prints.

OUTLOOK: **WHY ARE THERE TWO INCLUSION MECHANISMS?**

The main reason is that `link` is an HTML specific mechanism, but `@import` is a directive of CSS. This allows `@import` to be used in any CSS style sheet (e.g., also in XML style sheets). It also allows `@import` to be used inside external style sheets, e.g., to `link` to a single top-level style sheet which references several other style sheets rather than including all those links in every document.

Otherwise there is little difference, it is even possible to mix one or more linked and `@import` style sheets.

Due to deficient CSS support in various browsers, `@import` has been misused to hide some CSS rules from certain browsers. However, this practice is discouraged nowadays.[6]

. . . TO AN XML DOCUMENT

```
<Article>
  <ArticleTitle>An Introduction to CSS</ArticleTitle>
  <Author>Chris S. Smith</Author>
  <Paragraph>
    <Standard>CSS</Standard> can be used with <Standard>HTML</Standard> and
    with any structured document format, among other <Standard>YXML>/Standard>.
  </Paragraph>
</Article>
```

A style sheet must first specify which elements of the XML application considered are to be rendered *inline*, that is, not to be started or ended with line breaks at their ends, and which are to be rendered as *blocks*, that is, to start and end with line breaks. For the XML document given above, this can be done in CSS as follows:

```
Standard { display: inline; }
Article, ArticleTitle, Author, Paragraph { display: block; }
```

Element names in XML are case-sensitive. Therefore, selectors for an XML document are case sensitive, too. Therefore, the selector of the first CSS rule above could not be written `STANDARD`.

The CSS style sheet can be linked to the XML document using a **processing instruction** as follows:

```
<?xml-stylesheet type="text/css" href="lecture-note.css"?>
<Article>
  <ArticleTitle>An Introduction to CSS</ArticleTitle>
  <Author>Chris S. Smith</Author>
  <Paragraph>
```

```
      <Standard>CSS</Standard> can be used with <Standard>HTML</Standard> and
      with any structured document format, among other <Standard>XML</Standard>.
    </Paragraph>
  </Article>
```

EXAMPLE: **RENDERING XML DOCUMENTS WITH CSS**

The above rules together with the following rules

```
Article { color: black; background: white; font-family: "Gill Sans", sans-serif; }
ArticleTitle { font-size: 1.3em; }
Author { font-style: italic; }
Article, ArticleTitle, Author, Paragraph { margin: 1cm; border: solid; }
```

result in the following rendering of the XML document:

An Introduction to CSS

Chris S. Smith

CSS can be used with HTML and with any structured document format, among other

A.1.4.3 CSS STYLE SHEET AND STATEMENTS

A **CSS style sheet** is a list of *statements*. There are two kinds of statements: **"at-rules"** and **"rule sets"**.

At-rules start with a so-called "at-keyword", that is, the character `@` followed by an identifier (for example, `@import`, `@page`). At-rules end either with `;` or with a so-called **"declaration block"** (beginning with { and ending with }).

EXAMPLE: **AT-RULES**

```
@charset "ISO-8859-1";           /* encoding of a standalone style-sheet */
@import "print-main.css" print;  /* style-sheet import for the given media type */
@media print {
   body { font-size: 10pt; }
}
```

CSS distinguishes between the following mutually exclusive media types: all, braille, embossed, handheld, print, projection, screen, speech, tty, tv.

A rule set (short **"rule"**) consists a selector followed by a *declaration block*. A declaration block is a `;` separated sequence of declarations. A declaration consists of a property followed by `:` followed by a value (white spaces are allowed around each of property, `:` and value).

```
body { background: url("http://www.example.com/photos/sea-with-waves.png")
                   white;
```

```
3           color: black; }
```

GROUPING OF SELECTORS AND RULES

Both selectors and declarations can be grouped ("on both sides"):

- Instead of

```
1   h1, h2, h3 { font-style: bold }
```

  we can use

```
1   h1 { font-weight: bold; }
    h2 { font-weight: bold; }
3   h3 { font-weight: bold; }
```

- Instead of

```
1   h1 { color: blue; text-align: center; }
```

  we can use

```
1   h1 { color:blue; }
    h1 { text-align: center; }
```

Since font availability has been very different between different systems (e.g., Linux vs. Windows), CSS provides specifically for font specifications a means to specify alternative or *fallback* values:

```
    body { font-family: "Gill Sans", sans-serif; }
```

In this case, a browser first tries to use the font "Gill Sans". If that is not available, then the generic sans-serif font-family should be used (one of the generic five font families provided by CSS).

SELECTORS

A **selector** is either an *element name*, the character * meaning any element, or is composed as follows, where E and the E_i stand for element names:[7]

Combinators:	
$E_1\ E_2$	selects any E_2 which is a descendant of E_1
$E_1 > E_2$	selects any E_2 which is a child of E_1
$E_1 + E_2$	selects any E_2 which has an immediately preceding sibling E_1
$E_1 \sim E_2$	selects any E_2 which has any preceding sibling E_1
Attribute selectors:	
E[α]	selects any E which has an attribute named α (regardless of its value)
E[α="v"]	selects any E which has an attribute α with value v
E[α~="v"]	selects any E which has an attribute α whose value contains v as a word[8]
E#v	selects any E the `id` attribute of which is v
E.v	specific to HTML: selects any E with a `class` attribute whose value contains v as a word[8]
Pseudo-classes:	

[7] This list contains the most relevant selectors, for a full list of CSS 3 selectors see http://www.w3.org/TR/css3-selectors/#selectors.

E`:first-child`	selects any E which is the first child of its parent
E`:nth-child(`an + b`)`	selects any E which is the m-th child of its parent where $m = an + b$ for some n
E`:not(`S`)`	selects any E that does not match the (simple) selector S
E`:link`	selects any E which is the anchor of a not yet visited hyperlink
E`:hover`	selects any E over which the cursor hovers
E`:lang(`l`)`	selects any E in language l

Pseudo-elements

E`::first-line`	the first formatted line of an element selected by E

Recall that selectors can be grouped as in:

```
h1, h2, h3 { font-style: bold; }
```

PSEUDO-CLASSES The constructs beginning with `:` in the compound selectors refer to "pseudo-classes" (`:first-child`, etc.).

Only interactive media can support the following pseudo-classes:

```
a:link    { color: blue; }         /* unvisited links */
a:visited { color: red; }          /* visited links   */
a:focus   { background: yellow; } /* user hovers     */
a:active  { background: lime; }   /* active links    */
```

Pseudo-classes can be combined as follows:

```
a:focus:hover { background: yellow;}
```

PSEUDO-ELEMENTS Pseudo-elements start with `::` since CSS3 (in CSS 2 and CSS 1, they are prefixed with `:` and thus not syntactically different from pseudo-classes).

Specific rendering of the first line, or of the first letter, of a paragraph can be specified using pseudo-elements as follows:

```
p::first-line { text-transform: uppercase; }
p::first-letter { font-size: 200;
```

The `::before` and `::after` pseudo-elements are used to insert generated content (only plain text) before or after an element's content:

```
p.warning::before { content: "Warning: "}
```

An HTML p element like the following is rendered preceded by the text "Warning: ":

```
<p class="warning">
   Written examination on 12 February.
</p>
```

Pseudo-elements can only be used as the last expression in a CSS selector. E.g., `p::first-line > em` is **not** a valid CSS selector.

[8] A word is a sequence of non-whitespace, non-quote characters surrounded by either whitespace or quotes.

A.1.4.4 DETERMINING CSS VALUES

CSS is called *cascading* style sheet language to emphasize its ability to manage (possibly conflicting) property definitions from different sources: Style sheets may have three different origins: author, user, and user agent (that is, usually the browser).

The value of a property is then computed as follows (using the so-called *cascade*):

1. For the media considered, *sort* the declarations applying to an element after their *importance* (normal or important indicated by the `!important` directive in the CSS style sheet) and *origin* (author, user, or user agent) in ascending order of precedence as follows:
   a) user agent declarations
   b) user normal declarations
   c) author normal declarations
   d) author important declarations
   e) user important declarations

   The seemingly odd placement of author declarations between user declarations with normal and important priority is chosen so that by default the declaration of an author (i.e., Web page) override user declarations, unless the user specifically uses `!important` to indicate otherwise.
2. For a same element, declaration from rules with more specific selectors override declarations from rules with more general selectors.
3. If two declarations still can be applied to a same element, then the latter specified in the style sheet is retained. Declarations in imported style sheets are considered to be before any declarations in the style sheet itself.

THE FOUR VALUES OF A PROPERTY:

A property first has the value specified for it in the style sheet: This is its *specified value*. Such a value can be relative, for example `200%`, `3em` or a relative URI, and therefore cannot be directly used for rendering. Hence, an absolute so-called *computed value* is generated from a specified value. Computed values are determined before rendering.

Some rendering values cannot be determined before rendering because they depend on rendering sizes (for example of a parent element). Therefore, so-called *used values* are determined at rendering time from the corresponding computed values.

Finally, limitations of the rendering device might make approximations of some used value necessary. The values derived from the used values taking such approximations into account are called *actual values*.

INHERITANCE, OVERRIDING AND CASCADING

Consider the following HTML example and the associated style sheet:

```
      <p>Learning <em>CSS</em>is useful.</p>
2
      p { color: black;}
```

If no color has been assigned to the em element, the emphasized "CSS" will *inherit* the color of the parent p element. Thus, if p is assigned the (text) color of black, then the text of the em element will also be in bllack.

When such an inheritance occurs, elements inherit computed values: The computed value from the parent element becomes both the specified value and the computed value on the value inheriting child.

If however, the em element is assigned as follows the (text) color red, then the inherited value black is *overridden* and the text of the em element is in red:

```
p { color: black; }
em {color: red; }
```

The text in p (paragraph) elements is black except portion of texts in em (emphasized) elements that are red.

A few specific properties are not inherited for some (good) reasons:

```
body { background: url(hinterground.gif) white; color: black; }
```

white is an alternative value for the property background (for example in case the picture cannot be loaded by the browser).

background is not inherited. heading, paragraph, etc. elements nevertheless appear to have the same background as body because their default background is “transparent”. (This makes it computationally easy to achieve a good alignment of background images.)

Properties normally not inherited can be manually inherited using the value inherit.

EXAMPLE: **EXAMPLES OF CSS PROPERTIES AND THEIR VALUES**

CSS has about 90 properties. Only a few of those are mentioned in the following.

```
font-family: serif  /* or sans-serif */
font-style: normal  /* or italics */
font-weight: normal /* or bold */

font-size: 36pt
  /* no blank before unit, the points in CSS 2.1 are equal to 1/72nd of an inch) */
font-size: 432pc  /*1 pica = 12 points */
font-size: 1,5em
 /*width of the lowercase character "m" in the font used, also called quad-with */

line-height: 1,2cm
line-height: 120mm
line-heigth: 1.5ex  /* height of the lowercase character "x" in the font used */
```

CSS has two types of numbers: integers (like 23) and decimal numbers (like 23,567). Numbers of both types can but must not be signed (like -2 and +123,45)

CSS has a box model. A CSS box has, from the inside to the outside, a content area (for text or image) and optional surrounding padding, border, and margin areas. The size of each area can be specified by properties like the following:

```
margin-top: 1em
```

```
      margin-right: 0em
      margin-bottom: 1em
      margin-left: 0em

      margin: 2em          /* all margins set to 2em */
      margin: 1em 2em      /* top & bottom = 1em, right & left = 2em */
      margin: 1em 2em 3em /* top=1em, right=2em, bottom=3em, left=2em */
```

`margin` is a *"grouping property"* making compact specifications possible: The following

```
      margin: 1em 2em 3em
```

has the same meaning as:

```
      margin-top: 1em
      margin-right: 2em
      margin-bottom: 3em
      margin-left: 2em
```

CSS has 17 named colors (like blue, red, aqua, etc.). CSS uses the RGB color model for numerical color specifications. The following properties specify the same color:

```
      color: #f00               /* #rgb */
      color: #ff0000            /* #rrggbb */
      color: rgb(255,0,0)
      color: rgb(100%, 0%, 0%)
```

URIs can be referred to in different manners:

```
      background: url("http://www.example.com/photos/sea-with-waves.png")
      background: url(/photos/sea-with-waves.png)
         /* quotes are not mandatory, reletive URIs are allowed */

   li { list-style: url('http://www.example.com/blue-ball.png') disc; }
         /* properly paired single or double quotes are allowed */
```

Comments in a CSS style-sheet are enclosed between /* and */

A.1.5 HTTP and REST

HTTP, the Hypertext Transfer Protocol, is a request/response standard for distributed hypermedia systems realizing the **client-server** paradigm of distributed computing. Clients, called "user agent", issue requests using a web browser, crawler, or another tool, and responding servers, or "origin servers", deliver upon such requests resources such as HTML files they store or generate. In between the user agent and origin server may be several intermediaries, such as proxies, gateways, and tunnels.

A.1.5.1 HISTORY OF HTTP:

1989: HTTP, the **HyperText Transfer Protocol** to transmit web pages, is one of the three core ideas of Tim Berners-Lee initial conception of the web. (The two others were HTML and a platform independent web browser).

1996: Adoption of the pre-standard HTTP/1.1. by the major browser developers (Netscape, Mosaic, Lynx, Microsoft, etc.)

January 1997: Official release of the Request for Comments RFC 2068, the previous version of today's HTTP/1.1.

June 1999: Publication of the Request for Comments RFC 2616 [20] which defines HTTP/1.1, the version of HTTP currently in use.

HTTP/1.1 is a stable specification. The IETF/IEC httpbis Working Group has begun to revise HTTP/1.1. It defines its (humble) goal as follows:

> "The Working Group must not introduce a new version of HTTP and should not add new functionality to HTTP. The WG is not tasked with producing new methods, headers, or extension mechanisms, but may introduce new protocol elements if necessary as part of revising existing functionality which has proven to be problematic." `http://www.ietf.org/dyn/wg/charter/httpbis-charter.html`, September 2009.

HTTP is not constrained to using TCP/IP, although this is its most popular use via the Internet. Indeed, HTTP can be "implemented on top of any other protocol on the Internet, or on other networks" (quoted from [20], Section 1.4).

HTTP uses URIs for identifying data (called *"resources"*). **URIs (Uniform Resource Identifiers)** provide an identification mechanism "how to identify a document for retrieving/updating/linking". URIs have no semantics in the sense that any URI can be used for referring to anything. A URI is either a Uniform Resource Name (URN), or a Uniform Resource Locator (URL), or both. URIs are syntactically built as follows:

```
<scheme>:<scheme-specific-part>
```

Specific parts for the schemes `http` and `https` have the following form:

```
//<authority><path>?<query>?
```

The schemes specify a semantics: `http`, `https`, `mailto`, iTunes scheme, `gopher`, `urn`, etc.

EXAMPLE: **EXAMPLES OF URIS**

```
urn:isbn:0-486-27557-4
urn:issn:1535-3613
ftp://ftp.is.co.za/rfc/rfc1808.txt
http://en.wikipedia.org/wiki/URI#Examples_of_URI_references
mailto:mueller@ifi.unizh.ch
itms://phobos.apple.com/.../Results?artistTerm=U2&albumTerm=the%20joshua%20tree
gopher://spinaltap.micro.umn.edu/00/Weather/California/Los%20Angeles
```

HTTP is a *transport protocol for arbitrary data* in any internet media type (formerly called MIME types, with MIME being the acronym of Multipurpose Internet Mail Extension). Internet media type is an Internet standard that extends the format of e-mail to support:

- text in character sets other than ASCII
- non-text attachments
- message bodies with multiple parts
- header information in non-ASCII character sets

Through its **WebDav** (Web-based Distributed Authoring and Versioning) extensions, HTTP has become a full authoring protocol (for creating, editing, deleting, updating) for editing and managing files collaboratively on remote Web servers.

Typically, an HTTP client initiates a *request*. It establishes a Transmission Control Protocol (TCP) connection to a particular port on a host (port 80 by default). An HTTP server listening on that port waits for the client to send a request message. Upon receiving the request, the server sends back a status line, such as "`HTTP/1.1 200 OK`", and a message of its own, the body of which is the requested resource, an error message, or some other information.

HTTP has eight "methods" or **"verbs"**:

GET: *Requests* a representation of the specified resource. `GET` is "safe", that is, it should not be used for operations that cause side-effects (because documents retrieved using `GET` may be cached and `GET` may be used by robots or crawlers).

HEAD: Asks for the response identical to the one that would correspond to a `GET` request, but without the response body. `HEAD` is safe. (For retrieving meta-information in response headers, without having to transport the entire content. Used, e.g., to test whether a cached page has changed.)

POST: *Submits* data (in the `POST` request's body, e.g., from an HTML form) to be processed by the specified resource. May result in the creation of a new resource, or in updates of existing resources, or in both.

PUT: *Uploads* a representation of the specified resource.

DELETE: *Deletes* the specified resource.

TRACE: *Echoes* back the received request, so that a client can see what intermediate servers (proxies, gateways, tunnels) are adding to or changing in the request. `TRACE` is "safe".

OPTIONS: Returns the HTTP methods that the server *supports* for specified URL. This can be used to check the functionality of a web server by requesting * instead of a specific resource. OPTIONS is "safe".

CONNECT: Converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.

Safe HTTP methods are defined as side effect free: HEAD, GET, OPTIONS and TRACE. (This is not always enforced in practice: GET has side effects in many systems.)

Idempotent methods (repetition has no meaning) PUT and DELETE as well as all safe methods.

HTTP/1.1 has **persistent connections**. In HTTP/0.9 and 1.0, the connection is closed after a single request/response pair. Persistent connections can be used to speed several sequential request (e.g., HTML document, its CSS style sheet, contained images).

There are two ways of establishing a **secure HTTP connection:**

- the HTTPS URI scheme
- the HTTP 1.1 Upgrade header

Examples of HTTP transactions:

Retrieval request:

```
GET /path/file.html HTTP/1.0
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_5_6; en-us)
AppleWebKit/528.16 (KHTML, like Gecko) Version/4.0 Safari/528.16
Host: www.w3.org
Accept: text/html
Connection: Keep-Alive
```

Answer:

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/html
Content-Length: 1354
<html> ...
```

Update request with parameters "home" and "favorite flavor":

```
POST /path/script.cgi HTTP/1.0
User-Agent: HTTPTool/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 32
home=Cosby&favorite+flavor=flies
```

A.1.5.2 REST: REPRESENTATIONAL STATE TRANSFER

REST, the Representational State Transfer, is a **client-server architecture** for distributed hypermedia systems such as the Web which can, but must not, use HTTP.

A REST architecture makes of a network (for example of Web pages) a virtual *state machine*, allowing a user to progress through the application by selecting a link or submitting a short data-entry form. Each user action inititate a state transition and a transfer to the user of a representation of the new state.

Clients issue requests to servers; servers return appropriate responses. Requests and responses are built around the *transfer of "representations" of "resources"*. A **resource** is any coherent and meaningful concept that may be *addressed*. A **representation** of a resource is a *document* describing the current state of a resource.

At any time, a client is either transitioning between states or "at rest". A client at rest can interact with its user, but creates no load and consumes no per-client storage on the server(s) or on the network. The client sends a request when it is ready to transition to a new state. While one or more requests are outstanding, the client is in transitioning states. The representation of an application state contains links that may be used next time the client chooses to initiate a new state transition.

REST describes the interactions between the four *components* of the Web: origin servers, gateways, proxies and clients.

REST specifies six architectural *constraints* that so-called **RESTful system** must fulfill:

Client-server: Clients are not concerned with data storage, which remains internal to each server. Servers are not concerned with the user interface or user state.

Stateless: *No* client session context, or *session state*, is stored *on the server* between requests. Each request from a client contains all of the information necessary to service the request, and any state is held in the client.

Cacheable: Clients are able to *cache responses*. Responses must therefore, implicitly or explicitly, define themselves as cacheable or not cacheable to prevent clients reusing stale data in response to further requests.

Uniform interface: The communication between clients and servers is specified in an interface.

Layered system: A client *cannot ordinarily* tell whether it is connected directly to the end server, or to an *intermediary* along the way.

Code on demand (optional): Servers can temporarily to extend or customize the functionality of clients by transferring to them code to execute like Java applets or JavaScripts.

Web applications and servers that follow these principles, benefit in a number of ways: they are more easily scalable w.r.t. network bandwidth (by adding cache intermediaries) and server computation (since introducing new servers is easy as all state is contained in a request and thus no distributed state management is necessary).

On the other hand, most Web applications that go beyond serving static Web pages contain nowadays some form of server-side session management. Also the layered system principle (client cannot ordinarily distinguish between origin server and intermediaries) is in conflict with any application

that requires trust in the origin server (e.g., banking, shopping, etc.). In this cases, `https` is nowadays used which provides for origin server authentication.

For more details on the controversy around REST see [50].

The term Representational State Transfer (REST) has been introduced in 2000 with the doctoral thesis of Roy Fielding, one of the principal authors of the Hypertext Transfer Protocol (HTTP) specification versions 1.0 and 1.1.

Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. *ACM Trans. Internet Technol.*, 2(2):115–150, 2002 [21]. .

Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000. Chair-Taylor, Richard N [22]. .

Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. "big" web services: making the right architectural decision. In *Proc. Int'l. World Wide Web Conf. (WWW)*, pages 805–814, New York, NY, USA, 2008. ACM [39]. .

Stefan Tilkov. *REST und HTTP: Einsatz der Architektur des Web für Integrationsszenarien*. dpunkt Verlag, 2009 [49]. .

	d_1	d_2	d_3
algorithm	0	1/7	0
analysis	0	1/7	0
client	0	0	1/5
communicate	1/8	0	1/5
give	1/8	1/7	0
interface	0	0	1/5
investigate	0	1/7	0
name	1/8	1/7	0
network	1/8	2/7	0
serve	0	0	1/5
social	3/8	2/7	0
software	2/8	0	0
specify	0	0	1/5
use	1/8	0	0
usual	1/8	0	0

Table A.1.1: Term-document matrix

A.1.6 Basic Information Retrieval

Information retrieval (IR) is about *searching for textual documents*, for information within textual documents and for metadata about textual documents. Typical IR tasks are:

- find all *documents related to*, but not necessarily explicitly referring to, "computing" from a collection of document
- find all documents from a collection addressing *similar* issues as a given set of documents.
- Tell what a given *document is about* and whether it relates to issues addressed in a given collection of documents.

IR has been boost by World War II and the cold war: Immediately after World War II, the US military faced the problem of evaluating wartime technical and scientific documents captured from Germans; in the 50s of the 20th century, the fear of a "science gap" with the USSR led to developing mechanized literature search systems for the analysis of technical and scientific documents from the USSR.

Key to IR is the so-called **"vector space model"** developed by Gerard Salton in the early 1960s.

EXAMPLE: **DOCUMENT REPRESENTATION IN VECTOR SPACE**

Consider the following collection of documents:

d_1 : "Social software" is the name usually given to software for social uses like communication in social networks.

d_2 : "Social Network Analysis" is the name given to the algorithmic investigation of social networks.

d_3 : The communication between clients and servers is specified in an interface.

In a first phase called **"text normalization"**, each documents is transformed into a *set of words*. Article and some other auxiliary words not conveying much meaning by themselves are removed and the remaining words are reduced to their "stems", or base forms, each stem being associated an occurrence number:

d_1 : social/3, software/2, name/1, usual/1, give/1, use/1, communicate/1, network/1

d_2 : social/2, network/2, analysis/1, name/1, give/1, algorithm/1, investigate/1

d_3 : communicate/1, client/1, serve/1, specify/1, interface/1.

Then, a so-called **"term-document" matrix** is built, the columns of which correspond to the documents of the document collection, the lines to the stems, the entry given either their frequency, the so-called **"term frequency"**, (or, alternatively, their number of occurrences) of a stem in a document as shown in Table A.1.1.

In practice, hundreds or thousands of documents yielding each thousands of terms, as stems will be now referred to, are considered. In such cases, term-document matrices are *sparse*.

Instead of a term-document matrix, sometimes also the *document-term matrix*, its transpose, is considered where rows represent documents and columns terms.

A term-document matrix suggests to see a *document* as a (non-negative, real) *vector of terms*. A document not belonging to the collection considered can be reduced similarly to a vector of those terms occurring in the collection and one can measure its distance to the document vectors of the collection. A **query** Q can be similarly expressed as a vector v_Q of terms occurring in the collection, and those document vectors, or column vector of the term document matrix, closest to this vector v_Q can be returned as answer to the query, their distance to the query expressing to which degree they are answers.

Furthermore, any (non-negative, real) vector of terms can be seen as a "virtual documents". This way, a query can be considered answered by $2/3d_1 + 1/3d_3$.

Symmetrically, a *term* can be seen as a *vector of documents*, and queries over the terms can be expressed as row vectors and answered by those rows of the matrix that are close, or similar, to the query vector.

The basis considered for the vector space model is the standard basis even though its **orthogonality** often expresses an unrealistic *independence* among terms.

When the columns of a term-document matrix A and a query vector q are normalized, then the i-th components of the vector $q^T A$ expresses how well the i-th document matches the query. Indeed, the i-th component of $q^T A$ is the linear combination of q and the i-th column of A representing the i-th document, i.e. it sums up how well q is answered by the i-th document.

A.1.6.1 TF/IDF OR "TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY"

For performance reasons, other **weighting schemes** than *raw frequencies* are in general used. Variations from the model above often result in better performances. In particular, instead of term frequencies TF, the matrix entries often are TF × IDF where IDF, the so-called **"inverse document frequency"** or the inverse of the frequency of the term in the whole document collection. Several more sophisticated values are also used that are variants of the simple TF/IDF or term frequency-inverse document frequency. TF/IDF enhances the values assigned to terms specific to some documents and reduces the values of terms occurring in most documents.

A.1.6.2 VECTOR SIMILARITY BASED ON ANGLES/COSINES

The **similarity** of two documents, or of a query and a document, or of two queries, can be estimated as the *angle between the vectors* associated to the two documents, to the query and the document and to the two queries.

Indeed, in the vector space model, the norm (or length) of a vector can be seen as expressing how much a document repeats itself. The angle between two vectors instead expresses how close to each others the terms of the documents they represent are.

The similarity of a document (vector) v_1 to a document (vector) v_2 is measured as the cosine of v_1 and v_2:

$$\text{sim}(v_1, v_2) = \frac{v_1 \cdot v_2}{||v_1||_2 \cdot ||v_2||_2} = \frac{\sum_i (v_{1i} v_{2i})}{\sqrt{\sum_i v_{1i}^2} \sqrt{\sum_i v_{2i}^2}}$$

The relevance of a document (vector) d to a query (vector) q is measured as the cosine similarity of d and q.

The approach sketched above is called **"vector space model"**.

A.1.6.3 PERFORMANCE MEASURES

Two measures are commonly used for estimating the performances of IR systems: Precision and recall. Let Relevant_Q be the set of relevant documents for a query Q and $\text{Retrieved}_Q(S)$ the set of documents retrieved by IR system S for Q.

PRECISION: Precision is the fraction of the documents retrieved by the measured system S that are relevant to the query Q:

$$\frac{\text{Relevant}_Q \cap \text{Retrieved}_Q(S)}{\text{Retrieved}_Q(S)}$$

RECALL: Recall is the fraction of the documents relevant to the query Q that are successfully retrieved by S.

$$\frac{\text{Relevant}_Q \cap \text{Retrieved}_Q(S)}{\text{Relevant}_Q}$$

Both, high precision and high recall are desirable. In practice, precision is often improved at the cost of recall and vice versa. Both, high recall by low precision and low recall by high precision are questionable.

"False positive" is the name given to data wrongly returned as answers, "false negative" denote data that are answers but are not returned.

A.1.6.4 TEXT NORMALIZATION

Text normalization consist in the elimination of stop words and in stemming. Expressions like "The Rolling Stones" or "The Holly Father" that are better not reduced to stems make this phase much less trivial than it might seem at first.

Stemming can bne performed by brute force, that is, by look up in a table giving for each word its stem. Brute force stemming has two drawbacks: the table needs to be built, in the lack of algorithmic solution, by human work; a full coverage of a language is unlikely. (Some say look-ups might be too time consuming but I hardly believe it: a good index and/or data structure would make fast enough look ups possible.)

For the English language, stemming can also be performed *algorithmically* by suffix stripping rules like:

1. remove the ending -ly (example: badly → bad, foolishly → foolish)
2. remove the ending -ish (example: foolish → fool)
3. remove the ending -ed (example: occurred → occurr)
4. replace the ending -ies by -y (example: goodies → goody)

As the examples foolishly → foolish → fool suggest, rules may have to be applied one after the other to a same word. Some rule applications might be wrong, like embedded → embedd → emb). Several rules might be applicable to a same word. Rules can be completed by a table giving the stems in those cases that are not covered by the rules. A stem look up table can also be built from rules. Stemming often does not always generate stems that are words in the language (like for example in German lebe, lebst, lebt, leben → leb).

Lemmatization is more sophisticated than stemming and yields better results. While stemming applies to single words, lemmatization first isolate so called "parts of speech" like nouns and verbs, then applies to each kind of part of speech a specific stemming algorithm. (The traditional English grammar has eight parts of speech: noun, verb, adjective, adverb, pronoun, preposition, conjunction, and interjection (or exclamation).) In contrast to stemming, lemmatization avoids reducing both the noun "news" and the adjective "new" to the same stem.

Some stemming and lemmatization algorithms give the stem they generate for a word probabilities.

Text normalization is obviously language dependent. Algorithmic approaches convenient for a language might be much less convenient, or even impossible, for another language. While suffix stripping goes a long way for English, German requires both, suffix and prefix stripping (like in the example gelebt → leb). While English stemmers are relatively simple, the development of Arabic and Hebraic stemmers is still a research issue (among other reasons because in these two languages vowels are not written).

QUERY EXPANSION

Note that stemming is one of the techniques applied for so-called **"query expansion"**, that is enlarging a query vector with terms similar to those of the initial query. Others techniques used for query expansion are querying as well for synonyms of, for common false spellings of those of the initial query.

REFERENCES:

Amy N. Langville and Carl D. Meyer. Information retrieval and web search. In Leslie Hogben, editor, *The Handbook of Linear Algebra*. CRC Press, 2007. http://www.cofc.edu/~langvillea/HLA.pdf, checked 2009/10/25 [31]. .

Klaus U. Schulz. Information retrieval. Lecture notes, CIS, Ludwig-Maximilians-Universität München, 2004. http://www.cis.uni-muenchen.de/people/Schulz/IR_2009/IRmain.pdf, checked 2009/09/16 [46]. .

Amit Singhal. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–42, 2001 [47]. .

Kavi Mahesh. Text retrieval quality: A primer. Web page, Oracle Corp., 1999. http://www.oracle.com/technology/products/text/htdocs/imt_quality.htm, checked 2009/09/16 [34]. .

OUTLOOK: **VECTOR SPACES FOR THE MATHEMATICALLY TIMID**

First, *finite* vector spaces over the *reals* are sufficient to consider for IR and most computing applications. The intuition of such vector spaces it that of **coordinates** as we know them from maps: Objects called vectors are characterized by their coordinates.

The two basic operations of vector spaces are the *addition of vectors* and the *scalar multiplication* of a vector by a real number. Adding two vectors means pairwise adding their coordinates. Thus, if each vector represents the number of occurrences of terms in a document, then adding the two vectors yields the vector for the merge of the documents. If the vector components are frequencies, the sum of the two vectors is slightly more difficult to explain—but it can be. Try!

The scalar product of a document (query) vector by a real k can be seen as a strengthening of the document (query) making it, so to speak, to repeat itself k times.

Document (term) vectors are *linearly independent*, if their is no way to obtained one of them as a linear combination of the others, that is, each document (term) has a frequency of terms (document) that cannot be re-constructed by merging and strengthening other documents (term) from the collection. In other words, each document (term) contains information the others do not.

A *basis* of document (term) vectors corresponds to a set of documents (terms) such that every document (term) can be reconstructed from the basis, but no document (term) in the basis is redundant in the sense that it can be re-constructed from the others from the basis.

Orthogonal document (term) vectors are documents that refer to fully different terms (document).

Orthonormal document (term) vector bases are minimal sets of documents (terms) from which all documents (term) can be re-constructed, that pairwise do not have anything in common, and do not repeat themselves.

The *dimension* of document (term) vector space, that is the cardinality of one of its basis, is the minimal number of documents (terms) needed for re-constructing the whole document (term) collection.

A *norm* of a vector is a measure of its length. There are many ways to measure a length. Normalizing a document (term) vectors can be seen as making the document (term) not to repeat itself.

The **inner product** of two vectors is a way to measure the angles between the two vectors. Indeed, it is defined as follows:

$$x \cdot y = x_1 \cdot y_1 + x_2 y_2 + ... + x_n y_n$$

It is relatively easy to see on a drawing that in R^2:

$$x \cdot y = \cos(\text{angle}(x, y)) \cdot ||x||_2 \cdot ||y||_2$$

CHAPTER A.2

Applications: Search Engines and Network Analysis (PageRank and HITS)

A.2.1 Why are search engines needed?

Searching the Web for data is difficult for the following reasons:

- The Web has *no central organization*, it is self-organized, and therefore its evolution is hard to track.
- The Web is used for *all possible purposes* in all possible ways.
- Much Web data is *highly volatile*, i.e., it changes rapidly and is very much time dependent.
- Many actors on the Web act *selfish*: Some people try to *artificially* make some web data more visible on the Web (cf. invisible texts, misleading metadata, "link farms", "Google bombs", "spamdexing", etc.).
- The Web is huge (as of 2009: between $5 \cdot 10^{11}$ and 10^{12} web pages, i.e., one million millions).

In 1997 it became clear that standard IR methods were not appropriate for searching the Web. Two ideas emerged almost simultaneously, both using the link structure of the Web for its search:

- **HITS (Hypertext Induced Topic Search)** by Jon Kleinberg, then at IBM.
- **PageRank** by Sergey Brin and Larry Page, then at Stanford.

The basic idea of PageRank became the driving force (though not the only) behind the company Google started by Sergey Brin and Larry Page in 1998.

A.2.2 Components of a Web search engine

A Web search engine consists of the following four components:

Web crawler: "crawls" the Web for data (Web pages) and stores it in a directory.

It uses the structure index (see "Indexing module" below) for finding so far unknown/uncrawled data (pages).

Indexing module: extracts *keywords/phrases* and/or other *descriptors* from the data stored by the crawler. It puts the descriptors extracted in a *compressed* form and uses this compressed form for indexing Web data (Web pages) into

- a (text) content index (which is an inverted file),
- a (hyperlink) structure index.

To allow a *preview of the context* of keyword occurrences, the compressed form is also stored.

Query module: implements the *user query language* using the (content and structure) indexes and returns to a query the "relevant pages" as answer.

Ranking module: ranks the relevant pages.

The ranking modules is what *differentiates* search engines the most from one another.

The Algorithm known as PageRank (see Section A.2.3) is at the core of Google's ranking module. Google's ranking combines at least two scores:

- a *content* score,
- a *popularity* score.

The **content score** is query dependent and indicates how relevant the content of a page is for the given query. The precise algorithm used by Google is a trade secret. It takes into account over a hundred characteristics like font size, whether a keyword appears in a title, whether it is emphasized (`emph`, `b`, `i` in HTML), whether different keyword occurrences are close to each other in a page, and the content of neighbor pages.

The *popularity score* is mostly query independent and, in case of Google, based on the PageRank algorithm detailed in Section A.2.3. In addition to popularity derived from the structure of the Web, Google also uses popularity scores based on observed user behavior (e.g., which results have been clicked at often for similar previous queries).

OUTLOOK: **GOOGLE'S SUCCESS**

Three further aspects, beside the above mentioned software engineering (*content score*) and mathematics (*popularity score* PageRank) have played an essential role in Google's success:

- *Matrix computation techniques* resulting in an efficient implementation of PageRank (see Section A.2.3).
- *System engineering:* the engineering of a server network initiated by Sergey Brin and Larry Page before they created Google. Part of this is the development of a versatile programming technique for distributed computations (MapReduce, see Section A.4.1).
- A novel *business model for advertisements*: Keywords sold based on a combination of price bid and click-throughs, with bidding starting at US$0.05 per click. This model of selling keyword advertising was pioneered by `Goto.com`, a company created by Bill Gross, later renamed Overture and now owned by Yahoo!. The original model has been further refined by Google.

  In March 2009, Google began with "behavioral targeting" based on users' interests primarily mined from the set of web pages visited by a user (on which Google ads occur). This kind of interest mining has been first used by traditional Web advertisement companies such as DoubleClick (acquired by Google in 2007). In both cases, considerable criticism regarding potential privacy concerns have been raised, as most users are unaware of the practice and provided opt-out options are fragile.

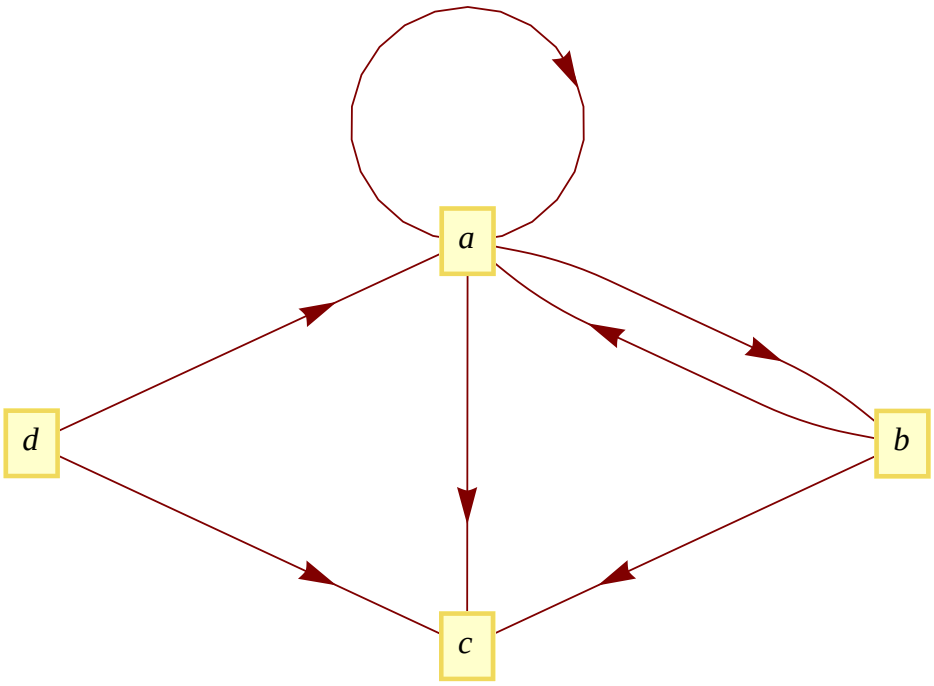


Figure A.2.1: Sample graph for PageRank computation

A.2.3 PageRank

A.2.3.1 ADJACENCY MATRIX

Consider the directed graph from Figure A.2.1 with four nodes (labelled a, b, c, d) and seven edges.

Its **adjacency matrix** A is shown below for the node order a, b, c, d. The cell a_{ij} in the i-th row and j-th column contains the number of edges[1] from the i-th to the j-th node, a 0 otherwise. E.g., $a_{23} = 1$ indicates that there is a single edge from b to c.

↗	a	b	c	d
a	1	1	1	0
b	1	0	1	0
c	0	0	0	0
d	1	0	1	0

The *sum of a row* of A is the number of outgoing edges of the corresponding node. The *sum of a column* of A is the number of incoming edges of the corresponding node. Any adjacency matrix is a square matrix, but symmetric if and only if for every edge (i, j) there is an edge (j, i).

[1] Here we consider graphs that allow multiple edges with the same start and end node, i.e., where the edges form a multi-set rather than a proper set. Obviously graphs with at most one edge with the same start and end node are a special case of such graphs.

Consider the Web as a graph, the nodes being the Web pages, the edges being the links between pages. The adjacency matrix of the Web is very sparse since each Web page links to a number of Web pages much smaller than the total number of Web pages.

A.2.3.2 HYPERLINK MATRIX

Assume a behavioral model of surfers where a surfer that is currently visiting page A *randomly* follows any of the outgoing links of A to get to the next page he is visiting.

The **hyperlink matrix** H of a set of Web pages $\{1,\ldots,n\}$ represents the probabilities to move from one page to another for such a surfer: h_{ij} indicates the probability that the surfer moves from page i to page j.

Let W be a Web graph with n pages (nodes) and $\mathsf{links}(j)$ denote the number of links from page j to any other page. Then $H = (h_{ij})$ is the **hyperlink matrix** for W, where

$$h_{ij} = \begin{cases} \frac{1}{\mathsf{links}(j)} & \text{if there is a link from page } j \text{ to page } i \\ 0 & \text{otherwise} \end{cases}$$

In other words, H is derived from the transpose of the adjacency matrix of the graph formed by links of the Web pages by dividing the nonzero entries of column j by links(j).

EXAMPLE: **HYPERLINK MATRIX**

For the graph from Figure A.2.1, the hyperlink matrix is

$$\begin{pmatrix} \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

The *sum of a column of H* is 1 if the corresponding Web page has at least one outgoing link, it is 0 otherwise. H is *not necessarily symmetric* since hypertext links are directed.

A.2.3.3 PAGERANK: PRINCIPLE

PageRank is based on the following *two basic ideas:*

- A page is important if it is linked to by other important pages. This is realized by defining the PageRank of a page P as the sum of the PageRanks of the pages linking to P.
- If page P links to several pages, then P's importance should be equally divided between the pages it links to.

If $\mathbf{x}$[2] is the "importance vector" (x_i is the importance of Web page i) for a Web with edges E, these two ideas can be expressed as the following *system of linear equations*

[2] In the following, we denote vectors like $\mathbf{x}$, though you might be more familiar with the notation $\vec{x}$.

$$x_i = \sum_{j:(j,i)\in E} \frac{x_j}{\text{links}(j)}$$

This system of linear equations can be expressed as follows *using the hyperlink matrix* H of the Web:

$$H\mathbf{x} = \mathbf{x} \tag{A.2.1}$$

A solution x of Equation A.2.1 is a so-called **eigenvector** of the matrix H (for the eigenvalue 1). Such an eigenvector has been called "conceptual PageRank" and gives, for each page i, its relative importance x_i.

Why does a solution of Equation A.2.1 represent a meaningful ranking of the web pages? H can be understood as propagating "importance" from web page to web page following the links. A vector $\mathbf{x}$ of web page rankings is the ranking sought for if it is stable under this propagation, that is, if $H\mathbf{x} = \mathbf{x}$, i.e., if Equation A.2.1 holds for that $\mathbf{x}$.

This raises the following question: When does a solution to Equation A.2.1 exist? This question must be re-phrased more precisely as follows:

- Is 1 **an eigenvalue** of the hyperlink matrix H? If not, Equation A.2.1 has no solution.
- Is there a **real (that is, non-complex) non-negative eigenvector** for the eigenvalue 1?

  Indeed, a vector $\mathbf{x}$ containing complex or negative real values could hardly be interpreted as a ranking.
- If such a vector exists, is it **unique**?

  If not, the approach would yield several different rankings and therefore be useless.
- If a unique, real, non-negative eigenvector for eigenvalue 1 exists, can we also compute it?

The following Section A.2.3.4 establishes the conditions that H must fulfill such that the first three questions have positive answers. In Section A.2.3.5, we then discuss how to transform an arbitrary hyperlink matrix, such that the conditions established in Section A.2.3.4 always hold on the transformed matrix (the so-called "Google matrix").

To answer the final question, we briefly outline the *power method* for computing matrix eigenvectors in Section A.2.3.6.

A.2.3.4 EXISTENCE OF A UNIQUE, REAL, NON-NEGATIVE EIGENVECTOR

The following theorem has been established by Perron in 1907:

Theorem 1 (Oskar Perron, 1907)**.** *Let M be a square matrix. If the entries of M are* ***strictly positive*** *(> 0), then,*

1. *there exists a (simple) eigenvalue $\lambda > 0$ such that $M\mathbf{v} = \lambda v$ and the corresponding eigenvector $\mathbf{v}$ of λ is real and* strictly positive *(i.e., all components of $\mathbf{v}$ are strictly positive, $v_i > 0$);*
2. *$\lambda = |\lambda| >= |\mu|$ for every other eigenvalue μ of M;*[3]

[3] Since λ is strictly positive $|\lambda| = \lambda$.

3. *any other strictly positive eigenvector of M is a multiple of v.*

A *simple eigenvalue* is an eigenvalue associated with only one eigenvector. The eigenvector of Perron's theorem is called **"Perron vector"**.

Perron's theorem is not convenient, since the matrix of the Web is sparse, that is, contains many zero entries.

Perron's theorem has been generalized in the following theorem, which is all that is needed to turn the conceptual PageRank into a unique, well-defined ranking of any Web:

Theorem 2 (Ferdinand Georg Frobenius, 1908-1912)**.** *Let M be a square matrix. If the entries of M are* ***non-negative*** (≥ 0) *and if M is irreducible, then*

1. *there exists a (simple) eigenvalue $\lambda > 0$ such that $M\mathbf{v} = \lambda\mathbf{v}$ and the corresponding eigenvector $\mathbf{v}$ of λ is real and* non-negative*;*
2. *$\lambda = |\lambda| >= |\mu|$ for every other eigenvalue μ of M;*
3. *every positive eigenvector of M is a multiple of $\mathbf{v}$;*
4. *if M has k eigenvalues of maximum modulus, then they are the solutions of the equation $\mathbf{x}^k - \lambda^k = 0$.*

A matrix M is irreducible if and only if it is the adjacency matrix of a strongly connected directed graph. A directed graph is said to be strongly connected if from each of its node, every other of its node is reachable along directed paths.

Frobenius' theorem generalizes Perron's theorem because if $M > 0$, then $M >= 0$ and M is irreducible. Since Theorem 2 generalizes Theorem 1, it is often called "Perron-Frobenius Theorem". The eigenvector of Frobenius' theorem is also called *"Perron vector"*.

Thus, all we need for answering positively the (first three) questions listed above and turn the conceptual PageRank into a flawless approach is to find a sensible way to tranform the hyperlink matrix H of a given Web W into an irreducible matrix (with positive entries) G. Indeed, if G is irreducible,

$$G\mathbf{x} = \mathbf{x} \tag{A.2.2}$$

has after the Perron-Frobenius Theorem 2 a unique, real and non-negative eigenvector, the "Perron vector", the associated eigenvalue of which is 1.

A.2.3.5 FROM HYPERLINK TO GOOGLE MATRIX

To transform the hyperlink matrix H of *any* given Web into an irreducible matrix (with positive entries) the following two easy steps suffice:

1. Construct from H the "**(column) stochastic hyperlink matrix**" H': A *column stochastic* matrix is a matrix where the sum of each column is 1. This is already true for all columns a hyperlink matrix H that have at least one outgoing link.

   For pages with no outgoing link (also called "*dangling* web page") the corresponding column is all 0. Therefore, we replace each such column with the vector $(1/n,\ldots,1/n)^T$ where n is the number of Web pages. (Instead, any non-negative stochastic vector could be used.)

We can also define H' using matrix operations: Let $\mathbf{d}$ be the vector of dangling pages such that

$$d_i = \begin{cases} 1 & \text{if } i \text{ is a dangling page} \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathbf{1}_n$ be the n-vector all components of which are 1. $\mathbf{d} \cdot \mathbf{1}_n^T$ is the matrix where the columns corresponding to dangling page contain only 1s, the other columns only 0s.

Then the column stochastic hyperlink matrix is defined as

$$H' = H + \frac{1}{n}\mathbf{d}\mathbf{1}_n^T$$

Each entry of H' is between 0 and 1 and the sum of each column of H' is 1, that is, H' is (column) stochastic (or markovian).

EXAMPLE: **STOCHASTIC HYPERLINK MATRIX**

For the graph from Figure A.2.1, the stochastic hyperlink matrix H' is

$$\begin{pmatrix} \frac{1}{3} & \frac{1}{2} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{3} & 0 & \frac{1}{4} & 0 \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{4} & \frac{1}{2} \\ 0 & 0 & \frac{1}{4} & 0 \end{pmatrix}$$

2. Construct the "Google Matrix" G from H': This step corresponds to turning the web into a strongly connected graph. Conceptually we thus add links (with very low weight) from each Web page to each other. These links are also called "random leaps".

   The Google matrix can be defined by matrix operations as well:

   $$G = (1-\alpha)H' + \alpha E$$

   where α is an arbitrary value between 0 and 1 and

   $$E = \mathbf{p}\mathbf{1}_n^T = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix} \begin{pmatrix} 1 & \dots & 1 \end{pmatrix} = \begin{pmatrix} p_1 & \dots & p_1 \\ \vdots & \vdots & \vdots \\ p_n & \dots & p_n \end{pmatrix}$$

   where where $\mathbf{p} = (p_1, \dots, p_n)^T$ is an arbitrary probability distribution, that is, $\mathbf{p} >= 0$ and $\sum p_i = 1$. The vector $\mathbf{p}$ is called *"personalization" vector* as it allows to adjust the "weight" of those web pages considered important (for example, those of universities).

   If not stated otherwise we assume that the random leap probability is the same for all Web pages, i.e., $\mathbf{p} = (1/n, \dots, 1/n)^T$ where n is the number of Web pages.

   α is sometimes called the "random leap factor" and it is often suggested that Google uses a factor of 0.15 (i.e., $1 - \alpha = 0.85$).

EXAMPLE: **GOOGLE MATRIX**

For the graph from Figure A.2.1, the Google matrix G for $\alpha = 0.15$ is

$$\begin{pmatrix} 0.320833 & 0.4625 & 0.25 & 0.4625 \\ 0.320833 & 0.0375 & 0.25 & 0.0375 \\ 0.320833 & 0.4625 & 0.25 & 0.4625 \\ 0.0375 & 0.0375 & 0.25 & 0.0375 \end{pmatrix}$$

Note that the Google matrix G is (column) stochastic (or markovian). 1 is always an eigenvalue of a (column) stochastic (or markovian) matrix.

The Google matrix is "primitive", that is, there is a $k > 0$ such that $G^k > 0$. For example, in the case of the Google matrix, $k = 1$ is such a value.

RANDOM WALKER: INTUITIVE INTERPRETATION OF g

G represents a a random walk through the Web: The random walker can be characterized as a Web surfer who, at each step, *randomly chooses a link* from his current page to click on except that:

- From a *dangling page* the walk is continued by jumping to another page selected at random (i.e., with probability 1/n);
- From each page, with probability α the walk can be continued by jumping to any page j with probability p_j (instead of following a link from this page). Thus a "random leap" (or jump) to page j occurs with probability $p_j \cdot \alpha$. Often p_j is $1/n$ where n is the number of pages and thus the probability of a "random leap" to j is α/n.

A.2.3.6 COMPUTING PAGERANK VECTORS: POWER METHOD

The **power method** is used by Google for computing the Perron vector of the Google matrix. The Perron vector of the Google matrix is commonly referred to as *"PageRank vector"*.

Applied to a primitive matrix and a positive vector, the power method always returns the *dominant eigenvalue and eigenvector* of that matrix, that is the eigenvalue satisfying (1) and (2) of the Frobenius' Theorem 2 and the eigenvector associated with it.

Algorithm 1 outlines the computation of the PageRank vector for a given Google matrix G with the power method.

A typical start vector is $\mathbf{e} = (1,0,\ldots,0)^T$ or any other standard base vector. Often a fixed number of iterations is used instead of an approximation quality ϵ.

Even though the Google matrix is completely dense (that is, none of its entries is equal to zero), the computations of the Power Method can be rewritten such that *only matrix-vector multiplication of extremely sparse matrices* and vectors are sufficient, which are significantly more efficient than multiplications with dense matrices:

$$G = (1-\alpha)H' + \alpha E$$

Algorithm 1: Computation of PageRank vector with power method

input : Google matrix G of a given Web with n pages
output: Vector $\mathbf{v}$, an approximation of the PageRank vector of G

1 **Initialization:**
2 $\mathbf{v}_0 \leftarrow \mathbf{0}$; *// null vector*
3 $\mathbf{v} \leftarrow$ arbitrary (real) non-negative n-vector;
4 $\epsilon \leftarrow$ arbitrary real between 0 and 1; *// approximation quality*
5 **while** $||\mathbf{v} - \mathbf{v}_0|| > \epsilon$ **do**
6 $\quad \mathbf{v}_0 \leftarrow \mathbf{v}$; *// save old value*
7 $\quad \mathbf{v} \leftarrow G\mathbf{v}$; *// power iteration*
8 $\quad \mathbf{v} \leftarrow \frac{\mathbf{v}}{||\mathbf{v}||}$; *// normalization*
9 **return** $\mathbf{v}$;

$$= (1-\alpha)\left(H + \frac{1}{n}\mathbf{d}\mathbf{1}_n^T\right) + \alpha\mathbf{p}\mathbf{1}_n^T$$
$$= (1-\alpha)H + \left(\frac{1-\alpha}{n}\mathbf{d} + \alpha\mathbf{p}\right)\mathbf{1}_n^T$$

Thus, instead of directly computing $\mathbf{v} \leftarrow G\mathbf{v}$ in Line 7 of Algorithm 1, we can compute

$$\mathbf{v} \leftarrow (1-\alpha)H\mathbf{v} + \left(\frac{1-\alpha}{n}\mathbf{d} + \alpha\mathbf{p}\right)\mathbf{1}_n^T\mathbf{v}$$

In this computation, the only product involving a matrix is $(1-\alpha)H\mathbf{v}$. Since H is very sparse (a Web page links to a very small number of Web pages), $(1-\alpha)H\mathbf{v}$ can be efficiently computed. It is estimated that a web page points in average to 10 pages. Thus, computing $(1-\alpha)H\mathbf{v}$ is in $O(10 \cdot n)$[4] given a suitable representation of H (e.g., as weighted adjacency list). It would be $O(n^2)$ if H would be dense.

Furthermore, the Power Method is *"matrix-free"*, that is, it uses matrices only in matrix-vector multiplications. This property makes a distributed computation possible, each sub-computation accessing only a part of the matrix. Since the Google matrix is $n \times n$ (with n equal to several billions), this is of considerable advantage. (More on this later in Section A.4.)

Furthermore, the Power Method does not require too much storage. It only needs to store:

- the very sparse matrix H,
- the (dense) dangling page vector,
- the (probably relatively sparse) personalization vector (most Web pages do not get a "personal" treatment),
- and, of course, the (dense) PageRank vector itself.

A.2.3.7 PROPERTIES OF PAGERANK

- PageRank, that is, the Perron vector of the Google matrix, is *query-independent*. Query independence makes it possible to keep Google's query answering fast as the Web grows.
- Query independence tends to skew the results in the direction of *importance* (measured by popularity) *over relevance* to the query: PageRank may have trouble distinguishing between

[4] Since n is a few billions, it is good to remember the factor 10).

pages that are authoritative in general and pages that are authoritative more specifically to the query topic. It is believed that Google engineers devote much effort to mitigate this problem, and this is where the metrics that determine the content score might have an effect.

- PageRank can be seen as a popularity contest with everyone on Web having a *vote*, that is, a social software (before the name was coined).
- PageRank was long believed to be relatively impervious to spam: it is easy to build link farms, but it is difficult to make them important enough for playing a role. However, with comment spam, Facebook spam, and spam placed on legitimate Web pages after security breaches, also Google has initiated strong anti-spam measures that penalize what Google considers "spam-like" behavior of Web pages (e.g., if the content of usually fairly stable pages changes rapidly, if a page contains many links and few text, if a page uses "hidden" text that is not visible to users, but to search engines).
- The "personalization" vector can be freely chosen: its choice affects neither mathematical nor computational aspects, but it does alter the rankings and the convergence speed (of the power method) in a predictable manner. This is of considerable advantage if Google wants to push a site's PageRank down or up (for example to punish a suspected "link farm").

REFERENCES

Klaus U. Schulz. Information retrieval. Lecture notes, CIS, Ludwig-Maximilians-Universität München, 2004. http://www.cis.uni-muenchen.de/people/Schulz/IR_2009/IRmain.pdf, checked 2009/09/16 [46]. Section 9.6 gives a very short and clear introduction to PageRank.

Amy N. Langville and Carl D. Meyer. Information retrieval and web search. In Leslie Hogben, editor, *The Handbook of Linear Algebra*. CRC Press, 2007. http://www.cofc.edu/~langvillea/HLA.pdf, checked 2009/10/25 [31]. Short and clear introduction to the mathematics of PageRank and HITS. There is a small mistake in the definition of H used in this chapter: On the third line from the top of page 14, it should be "from page j to page i" instead of "from page i to page j".

Pablo Fernández Gallardo. Google's secret and linear algebra. *Newsletter of the European Mathematical Society*, 63:10–15, 2007. http://www.uam.es/personal_pdi/ciencias/gallardo/google_ems_english.pdf, checked 2009/10/25 [23]. Clear introduction to the mathematics of PageRank with interesting side remarks, in particular on applications of the Perron-Frobenius theorems—for example to the rating of finance products by Standard & Poor's.

.

Amy N. Langville and Carl D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006 [30]. Clear book with interesting side remarks. Slides after the book on IR, vector space model, HITS and PageRank: http://www.cofc.edu/~langvillea/SMU.pdf.

Kurt Bryan and Tanya Leise. The $25,000,000,000 eigenvector: The linear algebra behind google. *SIAM Rev.*, 48(3):569–581, 2006. http://www.rose-hulman.edu/~bryan/googleFinalVersionFixed.pdf, checked 2009/10/25 [8]. .

PRIMARY ARTICLES

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. http://ilpubs.stanford.edu:8090/422/, checked 2009/10/25 [38]. .

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998. http://infolab.stanford.edu/~backrub/google.html, checked 2009/10/25 [7]. .

Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999. http://www.cs.cornell.edu/home/kleinber/auth.pdf, checked 2009/10/25 [29]. .

A.2.4 Hypertext Induced Topic Search - HITS

Like PageRank HITS uses the *hyperlink structure* for determining the "importance", or "popularity" of web pages. HITS differs from PageRank in two respects:

- The importance rankings HITS returns are query term dependent.
- HITS returns two instead of one importance ranking, one authority and one hub ranking.

Authority: a web page with many *incoming* links. The better an authority the more it is linked to. More precisely, a Web page is a "good" authority if it is referenced by many "good" hubs.

Hub: a web page with many *outgoing* links. The better a hub, the more Web pages it links to. More precisely, a Web page is a "good" hub if it references many "good" authorities.

HITS' intuition, like PageRank's, is expressed in a *recursive statement*: Good (that is, important) authorities are linked to good (that is, important) hubs and vice versa. This statement yields, like with PageRank's intuition, eigenvector equations.

Depending on the query, a *small set of Web pages* relevant to this query is built (e.g., by information retrieval techniques using keyword queries). This list is called **"root set"**. The root set is then extended with web pages that either (directly) point to pages in the root set, or are (directly) pointed to by pages of the root set. This yields the **"basis set"** with edges E_B.

Let $L = (l_{ij})$ denote a binary $n \times n$ adjacency matrix of the basis set, that is

$$l_{ij} = \begin{cases} 1 & \text{if there is at least one link from page } i \text{ to page } j \\ 0 & \text{otherwise} \end{cases}$$

Let a_i denote the **authority score** of page i, h_i denote the **hub score** of page i. Then, $a_i = \sum_{j:(j,i)\in E_B} h_j$ and $h_i = \sum_{j:(i,j)\in E_B} a_j$. Again, we can express these as vector-matrix multiplications:

$$\mathbf{a} = L^T\mathbf{h} \qquad\qquad \mathbf{h} = L\mathbf{a} \tag{A.2.3}$$

From these equations, one obtains by substituting **h** in the first equation by its value according to the second equation, and **a** in the second equation by its value after the first equation:

$$\mathbf{a} = L^T L\mathbf{a} \qquad \mathbf{h} = LL^T\mathbf{h} \tag{A.2.4}$$

Since $L \geq 0$, LL^T and $L^T L$ are ≥ 0 and symmetrical.

The symmetry of $L^T L$ can be proved as follows by referring to the graph considered: The entry l'_{ij} of LL^T is the number of paths from i over a link to a page P and from this page P following a link backwards to j. $l'_{ij} = l'_{ji}$ because every traversal of the afore-mentioned kind contributing from i to j yields a traversal of that kind from j to i.

The symmetry of $L^T L$ can also be proved algebraically as follows:

$$l'_{ij} = \sum_k l_{ik} l_{jk} = \sum_k l_{jk} l_{ik} = l'_{ji}$$

LL^T and $L^T L$ are also semi-definite, since $LL^T = (L^T)^T L^T = (LL^T)^T$ and $L^T L = L^T (L^T)^T = (L^T L)^T$ and their eigenvalues are all positive.

Solving the EquationsA.2.4 corresponds to finding the dominant eigenvectors of LL^T and $L^T L$. (Recall that the dominant eigenvectors are the eigenvectors corresponding to the eigenvalue with the largest absolute value.)

Since both matrices are positive semidefinite, the power methods applied with any vector converges and returns these dominant eigenvectors. Algorithm 2 shows the approximations of the vectors **a** and **b** for a given "base set" using the power method.

Algorithm 2: Computation of HITS authority and hub score with power method

input : Binary adjacency matrix L for HITS "base set"
output: Vectors **a**, **h**: authority and hub values of the Web pages in the "base set"

1 **Initialization:**
2 $\mathbf{a}_0 \leftarrow \mathbf{h}_0 \leftarrow \mathbf{0}$; *// null vector*
3 $\mathbf{a} \leftarrow$ arbitrary (real) non-negative n-vector;
4 $\mathbf{h} \leftarrow$ arbitrary (real) non-negative n-vector;
5 $\epsilon \leftarrow$ arbitrary real between 0 and 1; *// approximation quality*
6 **while** $||\mathbf{a} - \mathbf{a}_0|| > \epsilon$ *and* $||\mathbf{h} - \mathbf{h}_0|| > \epsilon$ **do**
7 $\quad \mathbf{a}_0 \leftarrow \mathbf{a}$; $\mathbf{h}_0 \leftarrow \mathbf{h}$; *// save old value*
8 $\quad \mathbf{a} \leftarrow L^T\mathbf{h}_0$; $\mathbf{h} \leftarrow L\mathbf{a}_0$; *// power iteration*
9 $\quad \mathbf{a} \leftarrow \frac{\mathbf{a}}{||\mathbf{a}||}$; $\mathbf{h} \leftarrow \frac{\mathbf{h}}{||\mathbf{h}||}$; *// normalization*
10 **return v** ;

It is also correct to use **a** and **h** on the right hand side of the assignments in Line 8.

REFERENCES

Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999. http://www.cs.cornell.edu/home/kleinber/auth.pdf, checked 2009/10/25 [29]. .

Chris Ding, Hongyuan Zha, Xiaofeng He, and Horst Simon. Link analysis: Hubs and authorities on the world wide web. *Siam Rev.*, 46(2):256–268, 2004. http://ranger.uta.edu/~chqding/papers/hits5.pdf, checked 2009/10/25 [17]. Analysis of properties of HITS.

Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, and Panayiotis Tsaparas. Finding authorities and hubs from link structures on the world wide web. In *Proc. Int'l. World Wide Web Conf. (WWW)*, pages 415–429, New York, NY, USA, 2001. ACM. http://www10.org/cdrom/papers/pdf/p314.pdf, checked 2009/10/25 [4]. Comparison of HITS and PageRank.

R. Lempel and S. Moran. Salsa: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst. (TOIS)*, 19(2):131–160, 2001. http://www.cparity.com/projects/AcmClassification/samples/383041.pdf, checked 2009/10/25 [33]. Combines the idea of a random walker with the hubs and authorities computation of HITS..

Klaus U. Schulz. Information retrieval. Lecture notes, CIS, Ludwig-Maximilians-Universität München, 2004. http://www.cis.uni-muenchen.de/people/Schulz/IR_2009/IRmain.pdf, checked 2009/09/16 [46]. Section 9.6 gives a very short and clear introduction to PageRank. There is a type at the end: Instead of $a = Ah$, it should be $a = A^T h$..

CHAPTER A.3

Query Languages: Keyword Query Languages

REFERENCES

Klara Weiand, Tim Furche, and François Bry. Quo vadis, web queries? In *Proc. Int'l. Workshop on Semantic Web Technologies (Web4Web)*, 2008 [52]. Slides for a tutorial based on this article can be found at http://pms.ifi.lmu.de/wise/.

Yi Chen, Wei Wang, Ziyang Liu, and Xuemin Lin. Keyword search on structured and semi-structured data. In *Proc. ACM Symp. on Management of Data (SIGMOD)*, pages 1005–1010, New York, NY, USA, 2009. ACM. http://www.cse.unsw.edu.au/~weiw/project/keyword_sigmod09_tutorial.pdf, checked 2009/10/25 [10]. .

CHAPTER A.4

Engineering: Inverted Files, Distributed Hash Tables, BigTable, and MapReduce for Network Analysis

A.4.1 MapReduce

Bibliography

[1] Ben Adida and Mark Birbeck. RDF/A Primer 1.0—Embedding RDF in XHTML. Internal draft, W3C, 2006.

[2] Oren Ben-Kiki, Clark Evans, and Ingy döt Net. Yaml ain't markup language (yaml) version 1.2. Informal specification, 2009. http://www.yaml.org/spec/1.2/spec.html, checked 2009/10/14.

[3] Paul V. Biron and Ashok Malhotra. XML Schema Part 2: Datatypes Second Edition. Recommendation, W3C, 2004. http://www.w3.org/TR/xmlschema-2/, checked 2009/10/14.

[4] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, and Panayiotis Tsaparas. Finding authorities and hubs from link structures on the world wide web. In *Proc. Int'l. World Wide Web Conf. (WWW)*, pages 415–429, New York, NY, USA, 2001. ACM. http://www10.org/cdrom/papers/pdf/p314.pdf, checked 2009/10/25.

[5] Tim Bray, Dave Hollander, Andrew Layman, and Richard Tobin. Namespaces in XML (2nd Edition). Recommendation, W3C, 2006.

[6] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and François Yergeau. Extensible Markup Language (XML) 1.0 (Fifth Edition). Recommendation, W3C, 2008. http://www.w3.org/TR/REC-xml/, checked 2009/10/14.

[7] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Comput. Netw. ISDN Syst.*, 30(1-7):107–117, 1998. http://infolab.stanford.edu/~backrub/google.html, checked 2009/10/25.

[8] Kurt Bryan and Tanya Leise. The $25,000,000,000 eigenvector: The linear algebra behind google. *SIAM Rev.*, 48(3):569–581, 2006. http://www.rose-hulman.edu/~bryan/googleFinalVersionFixed.pdf, checked 2009/10/25.

[9] Lillian N. Cassel and Richard H. Austing. ASN.1. Tutorial webpage, 2000. http://www.obj-sys.com/asn1tutorial/asn1only.html, checked 2009/10/14.

[10] Yi Chen, Wei Wang, Ziyang Liu, and Xuemin Lin. Keyword search on structured and semi-structured data. In *Proc. ACM Symp. on Management of Data (SIGMOD)*, pages 1005–1010, New York, NY, USA, 2009. ACM. http://www.cse.unsw.edu.au/~weiw/project/keyword_sigmod09_tutorial.pdf, checked 2009/10/25.

[11] James Clark. Associating Style Sheets with XML Documents, Version 1.0. Recommendation, W3C, 1999. http://www.w3.org/TR/xml-stylesheet/, checked 2009/10/14.

[12] John Cowan and Richard Tobin. XML Information Set (2nd Ed.). Recommendation, W3C, 2004. http://www.w3.org/TR/xml-infoset/, checked 2009/10/14.

[13] D. Crockford. The application/json media type for javascript object notation (json). RFC (Informational memo) RFC 4627, IETF, 2006. http://www.ietf.org/rfc/rfc4627.txt, checked 2009/10/14.

[14] Steve DeRose, Eve Maier, and David Orchard. XML Linking Language (XLink) Version 1.0. Recommendation, W3C, 2001.

[15] Steven J. DeRose and David G. Durand. The TEI hypertext guidelines. *Computers and the Humanities*, 29(3):181–190, May 2005.

[16] Steven J. DeRose and Andries van Dam. Document structure and markup in the FRESS hypertext system. *Markup Lang.*, 1(1):7–32, 1999. http://www.stg.brown.edu/~sjd/fress.html.

[17] Chris Ding, Hongyuan Zha, Xiaofeng He, and Horst Simon. Link analysis: Hubs and authorities on the world wide web. *Siam Rev.*, 46(2):256–268, 2004. http://ranger.uta.edu/~chqding/papers/hits5.pdf, checked 2009/10/25.

[18] Elika J. Etemad. Cascading style sheets (css) snapshot 2007. Working draft, W3C, May 2008. http://www.w3.org/TR/css-beijing/, checked 2009/09/16.

[19] David C. Fallside and Priscilla Walmsley. XML Schema Part 0: Primer Second Edition. Recommendation, W3C, 2004. http://www.w3.org/TR/xmlschema-0/, checked 2009/10/14.

[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. RFC (Request for Comments) 2616, IETF, June 1999. http://www.ietf.org/rfc/rfc2616.txt.

[21] Roy T. Fielding and Richard N. Taylor. Principled design of the modern web architecture. *ACM Trans. Internet Technol.*, 2(2):115–150, 2002.

[22] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. PhD thesis, University of California, Irvine, 2000. Chair-Taylor, Richard N.

[23] Pablo Fernández Gallardo. Google's secret and linear algebra. *Newsletter of the European Mathematical Society*, 63:10–15, 2007. http://www.uam.es/personal_pdi/ciencias/gallardo/google_ems_english.pdf, checked 2009/10/25.

[24] C. F. Goldfarb. Hytime: A standard for structured hypermedia interchange. *Computer*, 24(8):81–84, 1991.

[25] C. F. Goldfarb, E. J. Mosher, and T. I. Peterson. An online system for integrated text processing. In *Proc. American Society for Information Science*, volume 7, pages 147–150, 1970.

[26] Frank Halasz and Mayer Schwartz. The dexter hypertext reference model. *Communications of the ACM*, 37(2):30–39, 1994.

[27] Ian Hickson and David Hyatt. HTML 5: A vocabulary and associated APIs for HTML and XHTML. Working draft, W3C, August 2009.

[28] ITU-T. Abstract syntax notation one (asn.1): Specification of basic notation. Recommendation X.680, ITU-T, 2008. http://www.itu.int/rec/T-REC-X.680/en, checked 2009/10/14.

[29] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999. http://www.cs.cornell.edu/home/kleinber/auth.pdf, checked 2009/10/25.

[30] Amy N. Langville and Carl D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press, 2006.

[31] Amy N. Langville and Carl D. Meyer. Information retrieval and web search. In Leslie Hogben, editor, *The Handbook of Linear Algebra*. CRC Press, 2007. http://www.cofc.edu/~langvillea/HLA.pdf, checked 2009/10/25.

[32] John Larmouth. *ASN.1 Complete*. Morgan Kaufmann, 1st edition, 1999. http://www.oss.com/asn1/dubuisson.html, checked 2009/10/14.

[33] R. Lempel and S. Moran. Salsa: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst. (TOIS)*, 19(2):131–160, 2001. http://www.cparity.com/projects/AcmClassification/samples/383041.pdf, checked 2009/10/25.

[34] Kavi Mahesh. Text retrieval quality: A primer. Web page, Oracle Corp., 1999. http://www.oracle.com/technology/products/text/htdocs/imt_quality.htm, checked 2009/09/16.

[35] Jonathan Marsh. XML Base. Recommendation, W3C, 2001. http://www.w3.org/TR/xmlbase/, checked 2009/10/14.

[36] Jonathan Marsh and David Orchard. XML Inclusions (XInclude) Version 1.0. Recommendation, W3C, 2004. http://www.w3.org/TR/xinclude/, checked 2009/10/14.

[37] Jonathan Marsh, Daniel Veillard, and Norman Walsh. xml:id Version 1.0. Recommendation, W3C, 2005. http://www.w3.org/TR/2005/REC-xml-id-20050909/, checked 2009/10/14.

[38] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. http://ilpubs.stanford.edu:8090/422/, checked 2009/10/25.

[39] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs. "big" web services: making the right architectural decision. In *Proc. Int'l. World Wide Web Conf. (WWW)*, pages 805–814, New York, NY, USA, 2008. ACM.

[40] Steven Pemberton and the W3C HTML Working Group. Xhtml 1.0: The extensible hypertext markup language. Recommendation, W3C, 2000.

[41] Steven Pemberton and the W3C XHTML2 Working Group. Some early ideas for HTML. Information web page, 2009. http://www.w3.org/MarkUp/historical, checked 2009/09/16.

[42] Steve Pepper. The tao of topic maps. Tutorial webpage, ontopia, 2002. http://www.ontopia.net/topicmaps/materials/tao.html, checked 2009/10/14.

[43] Publications Office of the European Union. Interinstitutional style guide. Report, European Communities, 2009. http://publications.europa.eu/code/pdf/360300-en.pdf, checked 2009/10/14.

[44] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. Html 4.01 specification. Recommendation, W3C, 1999.

[45] Sam Ruby, Chris Wilson, Michael Smith, and Dan Connolly. W3C HTML working group. Home page, 2009. http://www.w3.org/html/wg, checked 2009/09/16.

[46] Klaus U. Schulz. Information retrieval. Lecture notes, CIS, Ludwig-Maximilians-Universität München, 2004. http://www.cis.uni-muenchen.de/people/Schulz/IR_2009/IRmain.pdf, checked 2009/09/16.

[47] Amit Singhal. Modern information retrieval: A brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24(4):35–42, 2001.

[48] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn. XML Schema part 1: Structures second edition. Recommendation, W3C, 2004. http://www.w3.org/TR/xmlschema-1/, checked 2009/10/14.

[49] Stefan Tilkov. *REST und HTTP: Einsatz der Architektur des Web für Integrationsszenarien*. dpunkt Verlag, 2009.

[50] Stefan Tilkov. Rest anti-patterns. InfoQ article, 2008 July. http://www.infoq.com/articles/rest-anti-patterns, checked 2009/09/16.

[51] Anne van Kesteren. HTML 5 differences from HTML 4. Working draft, W3C, August 2009.

[52] Klara Weiand, Tim Furche, and François Bry. Quo vadis, web queries? In *Proc. Int'l. Workshop on Semantic Web Technologies (Web4Web)*, 2008.

[53] Wikipedia. History of HTML. Wikipedia page, 2009. http://en.wikipedia.org/wiki/HTML#History_of_HTML, checked 2009/09/16.