A URL Classification System: Requirement Specification

1. Introduction

This document describes a system for classifying and storing the URLs of documents accessible via the World-Wide Web for later retrieval. The premise of the software is that once a user has discovered a useful web site, there should be an easy method of storing the URL for the site along with information which describes the contents in a form that facilitates later retrieval by various means. The software is different from a normal search engine in that the user has a role in describing and classifying the URL, and in that only URLs that the user has actually found useful are stored. Thus, subsequent searches on the database should yield more relevant URLs than a search on a general search engine.

The classification information for a URL will consist of a set of keywords along with a textual description of the document it refers to. The application should suggest keywords for the document by examining it for embedded meta-information provided by the document author as well as accepting new keywords provided by the user. It is assumed that there will be a companion application which allows the stored information to be queried and retrieved in various ways. While the characteristics of this companion application are not specified in this document, we must make certain assumptions regarding it in this specification. This companion application will be referred to as the query application. This application is described in a companion document.

2. Environment characteristics

2.1. Hardware

The application should run on both IBM-compatible and Macintosh hardware. We assume that the computer on which the application runs is connected to the Internet.

2.2. Operating System

While no information about particular operating systems was provided by the client, we assume that the application should run on the most recent end-user versions of the primary operating systems for each hardware platform. For the IBM-compatible system, this is Windows 2000. For the Macintosh, this is MacOS 9.1.

2.3. Other software

The system will need to interact in some way with a storage repository. The client has not specified any requirements on the nature of this repository, but since the stored information must be retrieved by the query application, it is natural to suppose that some fairly well documented and commonly available method for storing the data will be chosen. In the remainder of this document, we will refer to the mechanism used to store the URL information as the data store.
2.4. User expertise
The client is an experienced computer user. We therefore assume that users will have basic knowledge of using the mouse and keyboard, along with the typical text editing conventions for the two platforms mentioned in section 2.2, as well as basic knowledge of the World-Wide Web and web browsers.

3. Input/Output and System Modes

3.1. Data input

3.1.1. User input
We assume that data input will occur only via the keyboard and the mouse. The application will present text entry areas for the URL, the text description of the URL, and user-supplied keywords. The user should be allowed to select a subset of the keywords generated automatically by the application for inclusion in the data store.

3.1.2. Remotely retrieved data
Once the user has supplied a URL to classify, the application should discover as much information about the document as possible automatically. It should first check the existing entries in the data store and display this information if any is found. This allows editing of existing entries. If no information on the specified URL is found in the data store, then the application should attempt to retrieve the document via the Internet and scan it for relevant metadata. If no initial data can be found, the user will be presented with a blank form to fill in.

3.2. Data output
Data output consists of incorporating the displayed information into the data store for later retrieval.

4. Software functions

4.1. URL entry
The application will allow the user to enter the URL to be classified in some manner which allows full text editing and which will accept information pasted from other applications. Some action taken by the user once the URL is entered (e.g. clicking a button or selecting a menu item) will cause the initial attempt at automatic classification as described in section 3.1.2. to occur.

4.2. Automatic classification
Keywords generated through automatic classification of a URL should be displayed in such a way as to allow the user to specify which of the keywords to associate with the URL.
4.2.1. Retrieval of previously classified URLs
The application should be able to display previously classified URLs which are retrieved from the data store.

4.2.2. Use of metadata in classifying URLs
If no data for the requested URL exists in the data store, the application should attempt to retrieve the document specified by the URL via the Internet and to analyze it for potential keywords. The application should recognize relevant metadata supplied in any of the following formats:

1. Any META tag in the HEAD of the document with the name “keyword” or “keywords.”
2. Any keywords supplied using the Dublin Core syntax.
3. Any labeled keywords from the body of the document.
4. Any summary data supplied by the author, either in a META tag with the name “description” or in the equivalent Dublin Core syntax. This data can be used as an initial description of the URL.

4.3. URL information storage
When the user has added or amended the classification information for the URL, clicking a button or selecting a menu item should cause the information to be deposited in the data store.

5. Constraints and goals
(Note that the subsections of this section may vary from one project to the next; they are not set in stone.)

5.1. Portability
The application should be implemented in such a way that it can be used from either IBM-compatible or Macintosh personal computers. If possible, a common data store mechanism should be found.

5.2. Usability
The application should have a graphical user interface which makes its major functions easy to understand and to use. The appearance of the application on the two target platforms should be as close to identical as possible.

6. Response to undesired events

6.1. Inaccessible data store
If the application is unable to interact with the data store, it should notify the user and then exit.
6.2. **Internet communication failure**
In the case that the requested document cannot be retrieved from the internet, the user should be notified, but allowed to continue with the classification process.

6.3. **Invalid URL supplied**
In the case that the user specified a URL which does not conform to appropriate URL syntax, a warning should be issued and no attempt to retrieve the document from the Internet should be made.

7. **Life cycle considerations**

7.1. **Subsets**
1. Data entry subset - allow the entry of data about a URL without attempting to retrieve the document from the Internet for automatic classification. This subset would still be potentially useful.
2. Automatic document classification - retrieve and classify the document before allowing user modifications to the information about the URL.

7.2. **Fundamental assumptions**
1. We assume that the Internet and the World-Wide Web will continue to exist in something akin to their current forms.

7.3. **Potential changes**
1. Data storage formats or mechanisms may change.
2. Additional metadata formats may be invented and used.

8. **Glossary**

8.1. **Data store**
A mechanism used for long term storage and retrieval of arbitrary data.

8.2. **Document HEAD**
The portion of a web document between the `<HEAD>` and `</HEAD>` tags where metadata describing the contents of the document may be stored.

8.3. **META tag**
An HTML tag with may appear within the document HEAD and which is used to supply arbitrary information about the document.
8.4. **Metadata**  
Additional data appended to a document which describes the document’s content.

8.5. **Query application**  
A companion to the application described in this document which accesses the same data store and allows URL information to be retrieved by keyword.

9. **Sources**