A Knowledge Management Prototype

Mary S. Neff and James W. Cooper

IBM Thomas J. Watson Research Center

MaryNeff@watson.ibm.com, jwcnmr@watson.ibm.com

Background

As organizations become increasingly competitive, they recognize the capital inherent in the accumulated knowledge of its members. In a rapidly growing organization, it becomes ever more important to encapsulate and store this knowledge and invent ways to make it available to newer members of the team. We describe a Knowledge Management prototype, dubbed *Avocado*, a project that grew out of the need to provide sophisticated document analysis as an aid to retrieving more useful data from a technical document collection. It has evolved from earlier incarnations with different names (Cooper & Byrd, 1998; Neff & Cooper, 1999).

Avocado combines a number of components and sub-components, several of which have already become available in IBM products, for example, Intelligent Miner for Text. These components use, among other things, Natural Language Processing technology and corpus-based NLP techniques in the foreground and databases constructed using NLP technology in the background.

We have applied previous versions of the prototype to several collections of information technology (IT) documents and have developed a compact user interface for representing the results. Avocado represents a more ambitious effort in that it contains some new components and it is applied to a collection of "real data" that is more heterogeneous in form and content than any of our previous collections.

Avocado

The problem of finding important and relevant documents in an online collection becomes increasingly difficult as documents proliferate. Typically, searching for information in a document collection amounts to refining a query and then scanning a large list of documents returned by the search engine to determine their relevance, and then searching the documents for the desired information. Avocado's NLP and corpus-based NLP techniques assist the user both on the "query end" of the process, with Prompted Query Refinement (Cooper & Byrd 1997, 1998) and Lexical Navigation and on the "document viewing" end with Automatic Summarization, Keyword Extraction, and Active Markup (Neff & Cooper, 1999).

These several components rely on a number of more basic technologies, all of which run in the background when the documents are indexed for the search engine. They share the same data structures and work together to identify and index names, multiword terms, abbreviations, relations (named and unnamed) and count their frequency in the collection.

Prompted Query Refinement

Once the user enters a query, Prompted Query Refinement uses the index of names, terms, and relations (the collection vocabulary and the Context Thesaurus) to display to the user other related or possibly related terms that co-occur in the collection with the terms in the query. By selecting some of these related terms to be added to the query, the user can refine the query directly, without having to think up or type in any more items.

Lexical Navigation

In addition to the terms proposed by the Context Thesaurus, the system also retrieves terms from the collection database that are related to those terms in the query and can display the nature of the relations between these terms. These relationships can be both named relations ("CEO of," "makes," "is located in") (Byrd and Ravin, 1999) and unnamed relations, where terms have a strong bi-directional relationship. These relationships can be viewed in lists or plotted graphically.

Automatic Summarization and Keyword Extraction

For document summarization, we use a shallow summarization by sentence extraction method. Relying on the statistics of items in the document vocabulary and comparing the relative frequency of items in the document with that of items in the collection, we arrive at a notion of *salience* of items in the document. Terms in title and headings are also considered salient. The most salient items become the keywords. Sentences are scored according to the salience of the terms in them and their position in the document structure, and the most salient sentences (modulo a number of bonuses and exclusions based on discourse considerations) are extracted for a summary. A summary generated this way is not necessarily coherent, but we try to minimize this problem in the way that the summary is displayed and used (see Active Markup, below). In Avocado, we use a short, 4-sentence extract in the document hit list, and a longer one whose length is selected by the user in a frame at the top of the document when it is displayed.

Active Markup

Active Markup can be seen as a method of navigation through a group of documents. A document is displayed to the user together with an upper frame containing the list of most salient terms (highlighted in the summary and document by category or salience) and an extracted summary of the desired length. The keywords are active page components that can cause the server to return related information. In this implementation, the active components are used to query the server for a list of related terms to display. The related terms can then be used to compose another query for another list of documents. The summary sentences in the upper frame are hot-linked to their sources in the document, enabling the user to click to skip down to important information. This active markup approach coupled with the computer-generated summaries provides a form of "query-free" searching.

Implementation

The system is a Java client running in a frame of a web browser, which connects to a Java-based server running on Windows NT using Java RMI. This server in turn connects

to the database using JDBC and launches programs for carrying out the initial search and for producing the final summary as a pair of linked HTML documents which are displayed with the summary in an upper frame and the complete document in the lower frame.

In addition, a set of JavaScript form buttons labeled with the most salient keywords is displayed along the top of the document. Clicking on these form buttons launches a Java applet that can be used to display related keywords and the documents that contain them. This constitutes "Active Markup" of the document and provides an approach for query-free searching of the lexical neighborhood of the document.

Status

Avocado is a working system. At the time of writing, all the functions that we describe, as well as some others, have been integrated. Recently, we indexed the IBM Global Services consultants' reports on customer engagements. The data are much more problematic than what is found in well-edited news story or article formats. There are 50 large Lotus Notes databases, each for a different industry, with different editing, keyword, and submission criteria. Most of the documents have attachments in Word, WordPro, AmiPro, Freelance, PowerPoint, PS and PDF. Further, the interesting parts of the documents are the attachments. Not all the documents are in English, and some have no significant content (outlines, templates, management-speak). We plan to report on our experiences with this "real world" collection. A demonstration or ScreenCam movie will also be available.

References

Byrd, Roy J. and Ravin, Yael, 1999, submitted to NLDB99

Cooper, James W. and Byrd, Roy J. "Lexical Navigation: Visually Prompted Query Expansion and Refinement." Proceedings of DIGLIB97, Philadelphia, PA, July, 1997.

Cooper, James W. and Byrd, Roy J., OBIWAN - A Visual Interface for Prompted Query Refinement, Proceedings of HICSS-31, Kona, Hawaii, 1998.

Neff, Mary S. and Cooper, James W., ASHRAM: Active Summarization and Markup, Proceedings of HICSS-32, Wailea, Hawaii, 1999.