Automated Analysis of Structured Online Documents

Vladimir Kulyukin Kristian Hammond
Robin Burke
Artificial Intelligence Laboratory
Computer Science Department
The University of Chicago
Chicago, IL 60637
(312) 702-6209
{kulyukin, hammond, burke}@cs.uchicago.edu

Abstract

We believe that the domain-specific knowledge of the structural organization of information is central to the human ability to deal with large quantities of data efficiently. A better understanding of the computational nature of this ability may lead to solutions to information retrieval problems of practical significance. The paper outlines a Theory of Document Presentation (DPT) which addresses the problems of how information can be structurally organized in the documents of a given domain and how the standards for such organization emerge. The paper describes FAQ Minder, a document processing system whose implementation was guided by DPT. FAQ Minder processes FAQs, files of "Frequently Asked Questions" associated with USENET newsgroups [1,2]. The system identifies and tags the logical components of FAQs: network headers, tables of contents, sections, glossaries, questions, answers, and bibliographies.

1 Introduction

When the manager of a mutual fund sits down to write an update of the fund's prospectus, he does not start his job from scratch. He knows what the fund's shareholders expect to see in the document and arranges the information accordingly. An inventor, ready to register his idea with the Patent and Trademark Office of the US Department of Commerce, writes it up in accordance with the rules specifying the format of patent submissions. A researcher who wants to submit a paper to a scientific conference must be aware of the format specifications set up by the conference committee. Each of the above examples suggests that the domains of human activity that generate large amounts of documents are likely to have standards specifying how information is presented in those documents. Such standards, or *presentation patterns*, are a matter of economic necessity, not choice: documents whose visual structure reflects their logical organization are much easier to read and write than unconstrained text. We believe that the ability to *home in on* the needed content in the document by taking advantage of its structural organization allows people to deal with large quantities of data efficiently. Our contribution is based on the idea that the knowledge of the logical structure of a document[1] can be of significant assistance in processing the document's text.

[1]We use the terms *the logical structure of a document* and *the structural organization of information in a document* interchangeably.

2 DPT: What is the Problem?

DPT emerged from the ongoing research on FAQ Finder [1], an automated question-answering system whose main source of knowledge is the files of "Frequently Asked Questions" (FAQs) [2]. FAQ Finder answers natural language queries by matching them with the questions in its FAQs and returning the answer to the best match. Given a query, it selects a small number of probabilistically most relevant FAQs. Each FAQ in the FAQ library is a vector in a multidimensional space of terms selected by the SMART system [7]. Each question in a FAQ indexes the corresponding answer in a semantic net derived from Princeton's WORDNET. FAQ Finder matches the query against each question in the selected FAQs by passing markers in the net. It returns the answer to the question that scored the largest number of node intersections with the query.

When we started working on the question matching module of FAQ Finder, we were confronted with the problem of how to reliably identify questions, answers, tables of contents, glossaries and bibliographies in arbitrary FAQs. The problem is important because in order to answer questions about the contents of a FAQ it is useful to know its logical structure. For example, the frequent question *What is FAQ X all about?* can be answered by displaying X's table of contents. A good answer to the question *What books can I read about X?* is the bibliography of a FAQ about X. The semantic matching of questions and the retrieval of answers are impossible unless it is known which text regions of a FAQ are questions and which are answers. The problem is not trivial because neither parsing nor statistical word-based techniques are suitable for the job: state-of-the-art parsers are too limited in their scope of application; statistical methods mostly ignore the structural organization of text. DPT was developed as an attempt to find a comprehensive solution to the problem of how to identify the logical structure of a document.

3 DPT: Premises and Objectives

DPT is based on the following premises:

- **Economic Necessity Premise:** *The higher the demand for a class of documents, the higher the chances that the presentation of information in those documents will adhere to a small set of rigid standards.* Prospectuses of mutual funds, 10-Q forms, FAQs are but a few examples of document classes that emerged because of economic necessity and were standardized by consumer demand.

- **Text-As-a-Syntactic-Object Premise:** *As a source of information, text can be viewed as a syntactic object whose structural organization obeys certain constraints.* It is often possible to find the needed content in a document by using the structural organization of its text.

- **Presentation Consistency Premise:** *Document writers are consistent in their presentation patterns.* They do not change the chosen pattern within a single document; many of them stick with the same pattern from document to document.

- **Presentation Similarity Premise:** *The logical components of a document that have the same semantic functionality are likely to be marked in the same or similar fashion within a presentation pattern.* Document writers tend to mark headers, tables, sections, bibliographies, etc., in the same or similar ways in document texts.

4 Orthographic Markers and Their Sequences

The basic unit of the DPT formalism is the orthographic marker. Document writers use orthographic markers to specify the logical structures of documents; document readers use such markers to recognize those structures and home in on the needed content. Figure 1 gives a table with examples of some of the markers[2] found in many FAQs. Markers arrange themselves in marker sequences; sequences signal presentation patterns.

No	*Markers*	*Marker Types*
1.	**"Q:"**	alpha
2.	**"A:"**	alpha
3.	**"Subject:"**	alpha
4.	**"Section:"**	alpha
5.	**"[1-0]"**	alphanumeric
6.	**"1)"**	alphanumeric
7.	**"10.2a"**	alphanumeric
8.	**"VIII."**	alphanumeric
9.	**"**************"**	symbolic
10.	**"==========="**	symbolic

Figure 1: Markers and Marker Types

5 Constraints on Logical Structure

To accurately describe the logical structure of a document, it is necessary to reason about the marker sequences found in it: how they begin, get interrupted, restart, and end. The problem here is *sequence tracking*: how does the document reader, a person or a program, follow a marker sequence through the document. We have developed a set of constraints that specify how marker sequences behave in FAQs. FAQ Minder has four types of constraints. Each constraint is defined with respect to one of the three types of marker sequences. For lack of space, we give one example per constraint type:

Recursion Constraints: *Alphanumeric sequences do not recurse.* E.g.,

< *seq*1 > **[1] .... text .....**
< *seq*1 > **[2] .... text .....**
< *seq*1 > **[3] .... text .....**
..... text .....
< *seq*2 > **[1] .... text .....**

The recursion constraint forces the second "[1]" marker to start a new alphanumeric sequence < seq2 > and does not allow the first sequence < seq1 > to continue.

[2]We use the terms *markers* and *marker sequences* to refer to *orthographic markers* and *orthographic marker sequences.*

Crossing Constraints: *A symbolic sequence does not cross another symbolic sequence if the latter has started before the former.* E.g.,

< *seq*1 > **********
< *seq*2 > - ..... **text** .....
< *seq*2 > - ..... **text** .....
< *seq*1 > **********
< *seq*3 > - ..... **text** .....
< *seq*3 > - ..... **text** .....
< *seq*1 > **********

Here the crossing constraint forces the third "-" marker from above to start a new sequence seq3. If it belonged to seq2, seq2 would have to cross seq1, which the crossing constraint does not allow.

Stable Marker Structure Constraints: *Orthographic sequence do not change the structure of their markers.* E.g.,

< *seq*1 > **(1) ..... text .....**
< *seq*2 > **(1.1) ..... text .....**
< *seq*2 > **(1.2) ..... text .....**
< *seq*1 > **(2) ..... text .....**
< *seq*3 > **(2.1) ..... text .....**

In this configuration, the marker "**(1.1)**" signals a change of marker structure from **(lp integer rp)** to **(lp integer dot integer rp)**. The violation of this constraint often signals the start of a subsequence. In this case, seq1 has two subsequences - seq2 and seq3.

Proximity Constraints: *When the extenders of two different sequences with the same marker structure can extend on the same marker, only the sequence whose head marker is closer to the predicted marker is allowed to extend on it.* E.g.,

< *seq*1 > **A: ..... text .....**
< *seq*1 > **B: ..... text .....**
..... text .....
< *seq*2 > **A: ..... text .....**
< *seq*2 > **B: ..... text .....**
< *seq*2 > **C: ..... text .....**

The marker **"C:"** becomes an element of seq2 because its head marker **"B:"** is closer to **"C:"** than the same head marker of seq1.

6 LOGICAL MAP AND LAYOUT

Document writers frequently use markers in conjunction with *layout*. One way to compute the layout of a document is to digitize its text line by line according to a simple coding scheme. The coding scheme

implemented in FAQ Minder treats each line of a document as a string. Each substring of the string is mapped into the integer denoting its length. The special layout characters like *newline*, *tab*, and *space* are mapped into the integers 1000, 2000, and 3000, respectively. This scheme works because all FAQs in our library have lines that are at most 100 characters long. The database of assertions about sequences detected in the text of a document is referred to as its *logical map*. The presentation pattern of a document is determined by its logical map and its layout. As an example, consider the following chunk of text with the layout characters made visible:

<line 0 > *Tab Tab Space* **[1] first line.**
<line 1 > *Newline*
<line 2 > *Tab Tab Tab Space* **[1-0] second line.**
<line 3 > *Newline*
<line 4 > *Tab Tab Tab Space* **[1-1] third line.**
<line 5 > *Newline*
<line 6 > *Tab Tab Space* **[2] fourth line.**

The layout of the above text computed according to our scheme is:

(2000 2000 3000 3 5 5)
(1000)
(2000 2000 2000 3000 5 6 5)
(1000)
(2000 2000 2000 3000 5 5 5)
(1000)
(2000 2000 3000 3 6 5).

The logical map of the text follows. The symbols *lsb*, *rsb* and *hph* stand for *left square bracket*, *right square bracket* and *hyphen* respectively:

(alphanumeric-sequence seq1)
(occurs-on-line seq1 (lsb 1 rsb) 0)
(occurs-on-line seq2 (lsb 1 hphn 0 rsb) 2)
(occurs-on-line seq2 (lsb 1 hphn 1 rsb) 4)
(occurs-on-line seq1 (lsb 2 rsb) 6).

7 FAQ Minder: An Automated Document Processor

DPT has guided our implementation of FAQ Minder, a document processor which identifies the logical structures of FAQs, i.e. identifies and tags network headers, tables of contents, questions, answers, sections, glossaries and bibliographies. FAQ Minder works in two modes: supervised and unsupervised. In the unsupervised mode, the system takes as input the text of a FAQ and attempts to recognize its logical structure on its own. The current implementation deals only with tables of contents, questions and answers. In the supervised mode, FAQ Minder identifies the logical structure of the document by interacting with the user through a set of simple interfaces. Figure 2 shows the modules of FAQ Minder.

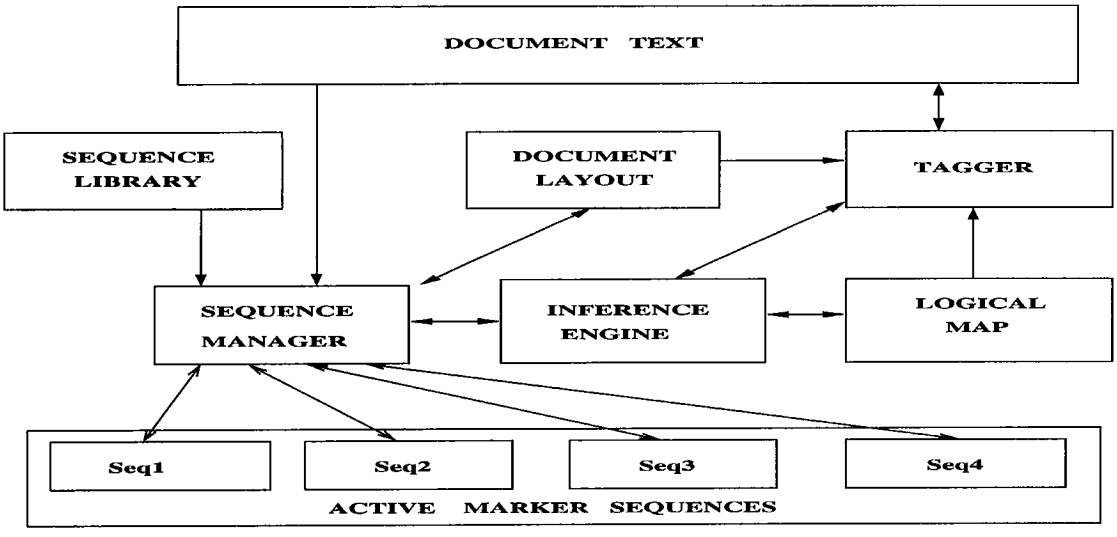

Figure 2: FAQ Minder's Modules

The Sequence Manager reads the text of a document, builds its layout and activates and deactivates marker sequences. The Inference Engine manages the logical map and the constraint satisfaction. The Tagger inserts tags into the text of the document given the document's layout and logical map.

FAQ Minder reads documents line by line. After a line is read, the Sequence Manager computes its layout, adds it to the document layout, and gives the currently active marker sequences a chance to examine the contents of the line. If a sequence recognizes a marker, it checks if the marker matches the layout of the previously recognized markers. The marker is ignored unless there is a layout match. When the sequence is finished with the line, it sends the Sequence Manager a message about its findings. If the sequence has recognized a marker and does not violate any constraints, which the Inference Engine checks against the logical map built thus far, the sequence is permitted to make the marker its current head and make the appropriate modifications in the logical map. If a constraint violation is inferred, the Sequence Manager puts the sequence on the list of inactive sequences. After the active sequences have examined the line, the Sequence Manager lets the previously deactivated sequences do the same. This second chance heuristic has its empirical justification in the fact that many marker sequences can start early in the document, get deactivated because of other sequences, and resurface in the middle or the end of the document. The library sequences are the last to inspect the line. After the layout and the logical map of the document have been computed, the Tagger uses them to tag the text of the document. Since it may need to do some additional inference, it is allowed to interact with the Inference Engine.

8 Discussion

We have tested FAQ Minder on 100 FAQs. Each of the FAQs in the experiment had two tags manually inserted in its text: one at the beginning and the other at the end of the text region containing the table of contents. FAQ Minder's job was to identify all of the entries in the table of contents and then,

using the information found in the table of contents, identify all the questions and answers in the rest of the document. All in all, FAQ Minder had to identify 2399 items.[3]

8.1 Evaluation Procedure

When FAQ Minder and a human judge were in agreement on an item, the item was said to be *completely recognized.* When FAQ Minder identified a part of an item, the item was said to be *partially recognized.* When FAQ Minder wrongly tagged a chunk of text as an entry in the table of contents, a question or an answer, that chunk of text was referred to as a *false positive.* The items identified by a human judge but missed by FAQ Minder were referred to as *unrecognized.*

8.2 Statistics

Of 2399 items 1943 items were completely recognized, 70 items were partially recognized, 81 were considered to be false positives, and 305 items were unrecognized. In percentage terms, FAQ Minder completely recognized 81 percent of items, partially recognized 3 percent, wrongly tagged 3.4 percent, and failed to identify 12.6 percent.

8.3 Causes of Failure

There were several causes of failure that we were able to identify while analyzing the results of the experiment. The most prominent among them was what we call *the phrasal change*: there was a considerable difference in how an table of contents entry and the corresponding question in the text were phrased. As an example, consider the following pair sentences:

[10] Where can I get APL software?
[10] Free Packages:

The first of these sentences is a table of contents entry; the second is the question that corresponds to it in the text of the FAQ about the programming language APL. The simple string similarity routine of FAQ Minder[4] failed to recognize the similarity in such cases. The second cause of failure was unknown orthographic markers. For instance, we discovered that many FAQ writers use Roman numerals which FAQ Minder could not handle. The third cause of failure was the difference between the orthographic markers in the table of contents and the orthographic markers in the text. For instance, *"20-2)"* and *"Subject: [20-2]"* were two different markers pointing to the same item in a FAQ. Several FAQs had numerous markers in the table of contents but none in their texts.

8.4 Future Work

Our future work on FAQ Minder will proceed in two directions. The first direction is to give FAQ Minder a limited ability to process natural language. Many FAQ writers state explicitly that a table of contents, a glossary or a bibliography is about to begin: *"Table of Contents"*, *"List of Topics:"*, *"The following questions are answered below:"* are but a few examples of sentences that precede tables of contents. If FAQ Minder can recognize such sentences, it can assume that the orthographic sequence that follows marks the table of contents. Toward this end, we have used the theory of Direct Memory Access Parsing[5,6] to specify 90 most common ways in which people can say that the table of contents

[3]Items in the experiment were entries in a table of contents, questions and answers.
[4]Two strings are similar if they satisfy the subset relation after being stoplisted.

is about to start. We have run our small program on 150 FAQs. The program correctly recognized such sentences in 111 FAQs. Our next step is to integrate that program into FAQ Minder. The second direction in which our research is likely to evolve is the supervised mode of FAQ Minder. Our conjecture here is that a system that recognizes its own limitations and overcomes them by asking the user intelligent questions saves the user more time in the limit than a completely independent system whose work requires laborious verification.

9 References

[1] Hammond, K.J.; Burke, R., and Schmitt, K. (1994). A Case-Based Approach to Knowledge Navigation. In AAAI Workshop on Knowledge Discovery in Databases. AAAI. August 1994. Seattle WA.

[2] Hammond, K.J.; Burke, R., Martin, C., and Lytinen, S. (1995). FAQ Finder: A Case-Based Approach to Knowledge Navigation. In AAAI Symposium on Information Gathering in Heterogeneous, Distributed Environments. AAAI. March 1995. Stanford University.

[3] Hammond, K.J. (1989). *Case-Based Planning: Viewing planning as a memory task.* Academic Press, Cambridge, Massachusetts, 1989.

[4] Schank, R.C. (1982). *Dynamic Memory: A theory of learning in computers and people.* Cambridge University Press, 1982.

[5] Riesbeck, C.K. and Schank, R.C. (1989). *Inside Case-Based Reasoning.* Lawrence Erlbaum Publishers, Hillsdale, New Jersey, 1989.

[6] Martin, C. (1991). Direct Memory Access Parsing. PhD Thesis, Yale University.

[7] Buckley, C. (1985). Implementation of the SMART Information Retrieval System. Technical Report 85-686, Cornell University.

[8] Salton, G. and McGill M.J. (1983). *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[9] Salton, G. and Buckley, C. (1991). *Global Text Maching for Information Retrieval.* Science, 253:1012-1015, 1991.

[10] Haralick, R.M. (1994). Document Image Understanding: Geometric and Logical Layout. 1994 Proceedings of IEEE Computer Society on

Computer Vision and Pattern Recognition. Seattle, Washington.

[11] Hearst, M.A. and Plaunt C. Subtopic Structuring for Full-Length Document Access. 1993 Proceedings of SIGIR. Pittsburg, PA.