A Graph-Based Information Integration Architecture

Jörg Caumanns
Center for Digital Systems
Free University of Berlin
Ihnestr. 24, 14195 Berlin
Germany

Abstract

In this research statement an information integration architecture is presented, that enables automated selection and sequencing of information within a given context. Each information providing object is described by domain-specific attributes and an generic index of required and provided keywords. By aggregating all available object's indices, a bipartite graph representing a structure of the knowledge domain is created. Various kinds of user requests can be answered by searching for certain subgraphs within the dependency graph. The approach presented is robust, generic, and well suited for adaptation and extensions as it does not rely on global semantic structures.

1 Introduction

Web search engines - such as *Altavista* - are perfect tools for finding all information related to a given topic. Nevertheless they are usually of no help for precisely answering a user's query as they cannot detect redundancy, do not provide adaptation, and are unable to integrate and structure the information discovered.

2 The i4 Framework

A generic framework for the automated, adapted, and structured integration of information is sketched in figure 1.

Starting point are various information repositories like the internet, databases, and textbooks. Information-encoding objects - e.g. images and paragraphs of text - can be extracted from these data sources by using web spiders, query languages, wrappers, OCR, etc. Information-encoding fragments from existing documents are called mediaobjects.

A preprocessing step is responsible for normalizing mediaobjects from different sources, e.g. by converting all texts into HTML. Further on some meta information about a mediaobject - language, color depth, size, etc. - can be retrieved during preprocessing as well.

Normalized mediaobjects are (manually) indexed and added to an information store which usually is a local or web-based database.

Given a set of normalized and indexed mediaobjects new documents can be created by restructuring and integrating this information. Within the i4-framework that was developed at the Free University of Berlin, integration and structuring of information is a 4-step process [Caumanns, 1998]:

First some measures are calculated on all mediaobjects. Then a subset of all objects available is selected according to a given request and a given user profile. The objects selected are sequenced and mapped onto a logical structure (e.g. chapters and sections). In a last step the structure of mediaobjects are converted into a user-readable document, which in case of the i4-framework is a set of hyperlinked HTML pages.

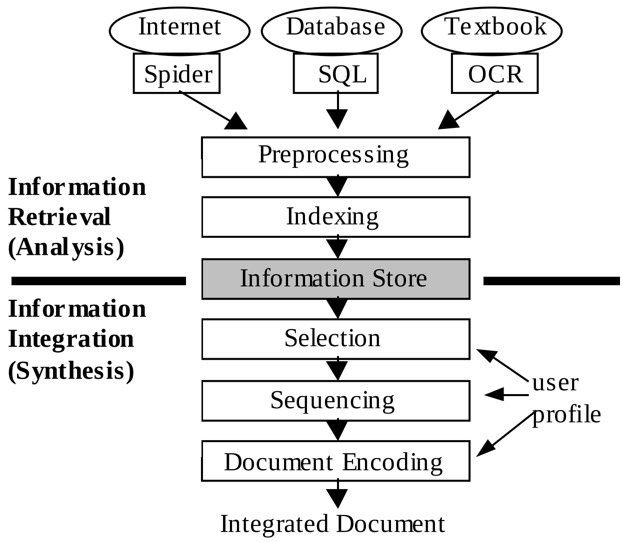

Figure 1: The i4 framework

* This work is supported by the German Research Society as part of the Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant no. GRK316)

3 Mediaobjects and Indices

The i4-framework uses a bipartite graph as its runtime-representation of the information store. Nodes of this graph - called the dependency graph - are mediaobjects and keywords.

Each mediaobject is described by a set of attributes. The number and kind of attributes depends on the application build upon the framework. Attributes can either be assigned manually or retrieved automatically. Most attributes are more or less transparent to the selection and sequencing algorithm. Their only purpose is to be used either as filters or as arguments for heuristic evaluations.

The content of each mediaobject is summarized by an index of all keywords either required or provided by the object. A keyword is said to be required if the mediaobject can only be understood if this keyword is known. It is provided if the mediaobject gives some amount of information about this keyword. Both - the amount of required and of provided knowledge - are weighted for each mediaobject-keyword mapping. These measures **can currently not be calculated automatically** with high accuracy and have to be provided (or at least checked) by hand.

A third content attribute - the position - defines how relevant a keyword is for a mediaobject. Automatically calculated values derived from frequency and position may be used for this measure.

4 Selection and Sequencing

Given these three kinds of content attributes, each mediaobject can be represented by a directed, bipartite graph. The nodes of the graph are the mediaobject itself and all its required and provided keywords. The edges of the graph are the mappings encoded within the mediaobject's index: Incoming edges start at required keywords while outgoing edges lead to provided keywords.

The aggregated graph of a set of mediaobjects is called a **dependency graph** as it reflects all dependencies between mediaobjects. Figure 2 shows a very simple dependency graph consisting of 9 mediaobjects (ovals) and 15 keywords (rectangles).

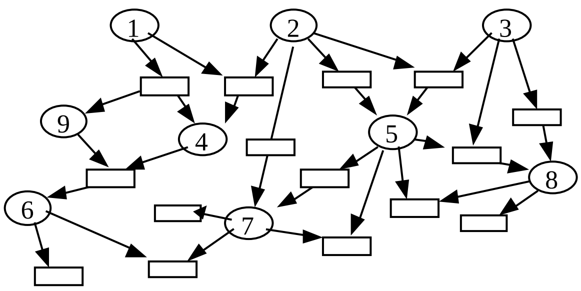

Figure 2: A simple dependency graph

To run a query against the dependency graph the following information must be given:

- the set of keywords the user wants to be informed about
- a user profile containing all the user's 'properties' with respect to language, bandwidth, hardware capabilities, prerequisite knowledge, etc., and
- the desired level of detail, structure (e.g. hypergraph or tree), level of redundancy, etc. for this specific query.

Based on this data the task to be performed can be defined as: Find a subset of the dependency graph within that all requested information is provided through a minimum set of linked mediaobjects that

- can be transformed into the requested document structure
- is adapted to the constraints and preferences given by the user's profile.

In order to find a close-to-optimum solution within reasonable time selection and sequencing of mediaobjects must be based on heuristics. To provide these heuristics with "global" information some measures are calculated for each node of the dependency graph and for the dependency graph as a whole. These measures include predecessor sets, tree structured subgraphs, and strongly connected components.

The algorithm used for selecting and sequencing the most appropriate mediaobjects is called a *Minesweeper algorithm* as it tries to implement the strategy human players use when playing this game. The Algorithm is based on the well-known A* algorithm [Nielsson, 1998], but tries to take the probabilities of mediaobjects for being part of the requested subgraph into account. This probability is the higher the worse all other providing objects for a single required keyword are. Selecting high-probability objects first, usually causes new high-probability objects to be selected next, which reduces the amount of 'vague' decisions to a minimum.

In a next step sequential paths of the calculated subgraph are collapsed. All mediaobjects within a path will be assumed to be placed on a set of sequential HTML pages. The mapping of mediaobjects to HTML pages is an NP complete optimization problem that is solved by an algorithm based on heuristics and try-and-error. In a last step the page structure is more or less 1:1 transformed into a logical structure consisting of chapters and sections.

References

[Caumanns, 1998] Jörg Caumanns. A Bottom-Up Approach to Multimedia Teachware. In *Proceedings of the 4th International Conference on Intelligent Tutoring Systems, ITS-98*, pages 116--125, San Antonio, Texas, 1998.

[Nielsson, 1998] Nils J. Nielsson, *Artificial Intelligence - A New Synthesis*, Morgan Kaufmann, San Francisco, California, 1998.