Compiling Document Collections from the Internet

V. Kluev
The Core and Information Technology Center
The University of Aizu
Tsuruga, Ikki-machi Aizu-Wakamatsu City, Fukushima, 965-8580
Japan
voice: [+81](242)37-2603; fax: [+81](242)37-2528
vkluev@u-aizu.ac.jp

ABSTRACT

Presently domain specific search engines are becoming popular because they offer greater accuracy, when compared to general purpose search engines. In this study, a method for collecting domain specific documents from the net was developed for the purpose of improving search results. The main thrust of our approach is to use several metrics to estimate the relevance of every automatically discovered document by a crawler regarding a topic of interest. This type of search resulted in two important findings. First, the time required for manual analysis of document content by the crawler was significantly reduced; second, the content quality of selected documents was improved. These results suggest that the rough estimation of precision and recall calculated in this study offer great promise.

Keywords

Search engine, data collection, relevance information, search context

1. INTRODUCTION

It is becoming clear that the Internet is the best source for obtaining information on virtually everything. With the explosive growth of the net, it is becoming increasingly difficult to find text documents on a particular topic. General purpose search engines like AltaVista, Google and Excite help only a little. There were a number of proposed approaches to improve this situation: 1) domain specific search engines [3], 2) focused crawlers [15] and 3) Web topic management systems [13]. All of these approaches are very similar to one another. The aforementioned search methods approach information retrieval using the following concepts. First of all, a set of documents (pointed to by the URLs) must to be specified. These documents collected by hand have to be relevant to the topic of interest. After that the crawler downloads documents from this set and creates (automatically or semi-automatically) a filter based on frequently occurring terms in these documents. The crawler then starts with a set of URLs and from there extracts other URLs which are followed recursively in a breadth-first or depth-first fashion. Documents that are relevant to the topic of interest are indexed. The following two considerations are critical in measuring the efficiency of the crawler: 1) How does the crawler make decisions about document relevancy (which technique is used)? 2) What is the order in which the URLs are traversed?

Some researches reported [3, 15] successful results for compiling thematic collections with high precision. However, it is intuitively clear that a crawler usually loses a great number of relevant documents. Recall is low. It should be noted this problem was discussed in [9] for filtering systems.

In this paper we try to address the question: Is there a more efficient and more effective way to compile a thematic (narrow) text document collections from the Internet? Our aims in these experiments are:

- to produce high precision results and to improve significant recall for them, and
- to compile collections in the shortest possible time.

The main thrust of our approach is the following. We propose to use several metrics to estimate the similarity of every discovered document to the topic of interest. We assume that a document is relevant undoubtedly, if it was appraised as relevant using two or more metrics. To evaluate documents we apply a combined boolean and vector space model metrics. In this case, the whole document may be estimated as non relevant to the topic of interest; however, such documents may actually contain some relevant information, which can be accessed using dynamic passage partitioning.

Some ideas applied here are taken from a number of different approaches to document retrieval [3, 7, 13, 15]. Our approach includes the following important features. Documents used to appraise and to index may be presented in HTML, PS, PDF formats as well as in plain text. Compressed files of text documents in formats like "zip" and "gz" are also evaluated. Files in other formats are not downloaded. Both the outgoing and incoming links are taken into account for relevant documents to the topic of interest. The AltaVista search engine assists our crawler in obtaining incoming links for essential documents. Only the first ten links are taken into account. All outgoing links for every document are extracted and added to a queue of links to visit. Every link inherits an average arithmetic sum of the score from its parent documents. The queue is arranged according to score of its elements in decreasing order. To bring the queue up to date, elements are moved after re-calculating its score. Automatically detected relevant documents are selectively checked by human inspection.

The paper is organized as follows. Related work are discussed in section 2. The OASIS system in which the proposed approach incorporated is briefly described in section 3. The method used for crawling the net is presented in section 4. Results of crawling are discussed in section 5. Final remarks are provided in section 6.

2. RELATED WORK

There is a large body of literature related to domain-specific (focused) crawlers and search engines. Extensive spidering is the key to obtaining high coverage by general purpose search engines (AltaVista, Google, etc). Since the aim of these systems is to provide search capabilities over the whole net, they simply try to discover as many distinct pages as possible. The strategies like breadth first search suits for this purpose.

A domain specific search engine can be defined as an intelligent spider, which should avoid links (URLs) that lead out of topic areas, and concentrate on links that lead to documents of interest [3]. Many such domain specific search engines have gained popularity as a way to find information on the Internet. Since building these search engines is a labor intensive process, requiring significant human effort, a semi automatic construction is a promising alternative.

Some of the proposed systems are oriented toward collecting a narrow class of topics and using specific heuristics. The Cora [3], for example, is designed to collect computer science research papers in PS format. Another tool is designed for searching a program's source code [6]. An interesting approach to find and analyse business information from the Internet is presented in [8].

The second class of such systems is more universal. The tool can be tuned for a domain. Descriptions of these systems can be found in citations [13, 15]. The most popular techniques used to determine the relevancy of documents are a probabilistic model [11] and a vector space model [7]. These techniques are usually combined with a number of heuristics. The heuristics are designed to select the most relevant links to visit and estimate, and to find special features in documents which are suppose to be relevant to the topic.

The vector space model was used in the system [13]. The researches concluded that result collections of HTML documents, which were built using this metric, were a good source of information available on the net for the user–specified topic. We argue against this point. As research [16] shows: Results depend very heavily on the topic.

Methods of crawling the Web for domain-specific information are discussed in citations [3, 6, 8, 9, 13, 15].

The aforementioned system, Cora, determines which documents are reliable based upon whether or not the documents have a research format (e.g. by having Abstract and Reference sections). The main idea of another heuristic is the following. Researchers discovered that majority of pages comprising computer science department web-sites contain links to courses, homework and schedules, etc, rather than links to research. So, avoiding whole branches and neighborhoods of department web graphs can significantly improve efficiency and increase the number of research papers. The system proposed in [15] considers every document as a bag of words. Researches applied a probabilistic measure to compute the similarity of documents. A combination of a vector space model with an artificial neural network is used in the filtering system described in [12]. The system from [8] uses different agents for crawling the WWW, evaluating and visualizing the results. The crawling agent works like a metacrawler collecting results of different search engines. The simple term frequency model was implemented in the ranking agent. Before visualization, documents are divided into groups by a clustering agent. An approach to filter documents using a non relevant information profile was tested in [9]. Researchers found that many non relevant documents are relatively close in similarity to that of relevant documents. The non relevant information profile express the features of mistakenly retrieved non relevant documents. The aim of this profile is to discard the retrieval of non relevant documents which are similar to documents previously mistakenly retrieved. In the tests the similarity to the non relevant information profile was calculated, and documents with high score were discarded.

Additionally, the idea of a distributed search is getting more popular. One of the reasons for this is an impossibility to create an index of the whole net on one computer due to the limitations of hardware and network resources of any given server. Approaches to construct distributed search system are discussed in citations [4, 7]. The OASIS [1] system offers a promising solution of a distributed search. This system is briefly discussed in the next section.

3. THE ARCHITECTURE OF THE OASIS SYSTEM

OASIS (Open Architecture Server for Information Search and Delivery) was developed by an international consortium in the framework of the INCO Copernicus program of the Commission for the European Communities (Project PL 1116-96, INCO Copernicus, Framework IV) in 1997 - 1999. The aim of the project was to develop an open architecture of a distributed scalable network of the Internet search servers, which provide intelligent search services for plain text and HTML documents, publicly accessible on the Internet through HTTP and FTP protocols. The tight cooperation of academic and industrial partners from four countries was a key factor to make this project successful. The partners are: St. Petersburg State University (Russia), the University of Tuebingen (Germany), the University College Dublin (Ireland); and commercial companies: Peterlink (St.Petersburg, Russia), Valtek Ltd. (Ukraine) and DSI Ltd. (Russia). The system is an open source software. It can be obtained from http://www.oasis-europe.org.

The basic idea of the OASIS approach is the following. The OASIS service presents a distributed system of search engines in the Internet. The typical OASIS server creates and supports one or more subject-specific indices. The user query is only propagated to a subset, which possibly contains the requested information. After merging, returned results are presented to the user in the usual search engine manner. The configuration of the OASIS server without any indices is possible.

The main obligatory components of the OASIS server are a Query Server (it manages distributed query processing) and zero or more Crawlers (they assist in building and refreshing the server's indices). Responsibilities of the Query Server include query propagation to a set of Collections in the OASIS service and search results merging. In addition to query propagation, the Query Server also propagates relevance feedback to all OASIS servers which have taken part in the distributed search. This feedback can be processed by the servers to improve subsequent query processing. Collections in the system are not required to be mutually exclusive in terms of the topic areas they cover. This facilitates individual, and possibly competing, proprietorship of OASIS servers. Every Collection must provide a standard interface to participate in OASIS. The aim of the OASIS Crawler is to maximize the relevance of the document records in its Collection with respect to the Collection's topic. Crawlers can cooperate with each other for the improvement of local Collection quality propagating discovered URLs. The quality of collected Crawler documents defines the quality of Collections in the OASIS system.

4. CRAWLING METHOD

The main feature of our method is to use several metrics to estimate the similarity of a discovered document regarding a topic of interest. These metrics are as follows:

- *a word collocation metric*
- *a term metric*
  A document is considered as relevant, if the number of three or two word collocations or terms from the filter, which appear in a document, dependent upon the threshold set for collocations and terms respectively.
- *a vector space model metric applied to the whole document*
- *a vector space model metric applied to dynamic passage parts of the document*
  The dynamic passage partitioning is done as follows. The window in 200 characters is shifted though the document with a step with 50 characters. Every selected passage is considered as a whole document.
- *a metric on the base heuristic*
  If one relevant document is found in the directory, all documents from the same directory are also relevant (these documents are non relevant in aforementioned metrics).

All thresholds must be tuned during training the crawler on the document core.

The aforementioned most popular idea to crawl the net was applied in the following fashion.

1. M documents, which are relevant to the topic of interest, have to be collected from the Internet by a human expert in this field. HTML, PS, PDF, plain text formats are possible. Compressed documents of these formats can also be selected. It is better to collect them from different sources. An amount of $100 - 150$ is enough to determine significant terms and word collocations. Selected documents form a document core.
2. Word collocations before an elimination of stop words are automatically formed, and then terms are extracted from documents. In the case of necessity, documents are unpacked and converted into plain text. HTML tags are discarded. Methods of the word collocation constructions are discussed in [11]. We use the simplest approach: collocations are built from words not separated by stop words or punctuations. Experiments were conducted with two and three word collocations.
3. The permanent part of a filter is constructed from lists of pairs "term term_frequency", "collocation number_occurrence" (consisting of two and three words) by a human expert. His task is to select the most significant elements. The selection range should be approximately between 20 and 100 from each class.
4. The estimation of a threshold for the filter is made during an evaluation of the document core by the crawler. (The structure of the filter and the threshold will be discussed below.) The crawler has to recognize about $60 - 70\%$ of documents from the core as relevant to the topic of interest.
5. The crawler runs and collects information for a particular topic until the the queue of URLs to visit is empty or a user-specified limit is reached. The idea of tuning the crawler filter after a recommended number of documents is useful [10] and it is applied here in the same style. The crawler filter consists of permanent and variable parts. The variable parts are changed every time upon the inclusion into a recommended amount of N documents. This number of documents are considered as a core. Operations explained from step 2 are conducted for these documents. Components for the variable part of the filter are randomly selected from a middle frequency. A number of elements in each class is equal to 30% of the total amount of a persistent part. Weights of all terms, which are used in the vector space model metric, are corrected with a damping factor. It is set to 0.2. At this point, the crawler constructs:
   - a database of visited URLs to avoid revisiting them,
   - a database of recommended documents,
   - a database of URLs to be visited.

   Of the three databases shown above, the first is cleaned after every evaluation of L documents. The aim of the second is to avoid a recommendation of documents, which were previously recommended. The third database presents a source of URLs to appraise documents. The standard tools are used to unpack compressed files and to convert PS and PDF files into plain text: pstotext [5], pdf2ps, gunzip, unzip.
6. Document–duplicates downloaded from different servers ("mirrors") can be recognized by hashing an incoming document and matching it against existing documents. This technique, used in Excite [7], is effective.

Since the same documents in different formats cannot be selected using this technique, we adopted a method proposed in [2]. Two documents A and B are considered as the same, if the following equation is above an equivalency threshold (we used the value 0.9 for the threshold in our experiments).

$$r(A,B) = \frac{|S(A) \cap S(B)|}{|S(A) \cup S(B)|}$$

Where $S(A)$ is a set of all unique subsequences consisting of 5 words taken from the document A, $|A|$ is the cardinality of set A. Since this approach has a very high computational complexity, it was used for documents with a length difference of less than 500 characters (the length was calculated after elimination HTML formatting and URLs from documents). We also applied a very simple method. If names of document files are differed only in their suffixes we assumed that they are the same. This generally is effective.

7. Documents recommended by the crawler need to be evaluated by human experts. However, it is impossible to do this for thousands of documents the crawler retrieved. We use the following methodology to reduce the amount of manually categorized documents: Documents retrieved from one directory were placed in a corresponding one. Special subdirectories were made for documents which were categorized as relevant:

   - using a word collocations and term metric.
   - using a vector space model metric for the whole document.
   - only using a vector space model metric for dynamic passage partitioning.
   - using at least two aforementioned metrics.
   - only using the metric on the base heuristic mentioned in the beginning of the section.

   We assumed, that documents from the fourth subdirectory are undoubtedly relevant. The manual evaluation for documents from other subdirectories is done as follows. The expert randomly selected and then categorized 20% of documents from each subdirectory. If more that 75% of them were relevant to the topic of interest, then we assumed that all documents in the directory were also relevant except for manually detected non relevant ones. In other case, (number relevant documents was less than 75%) we supposed that the directory included mostly non relevant documents and left only relevant ones from the evaluated set; all other documents were deleted.

5. RESULTS OF CRAWLING THE WEB

It should be noted that experiments conducted with files located in the Internet are very difficult, because it is practically impossible to repeat them in the same style. The reason is that the net is changing continuously. Moreover, the crawl depends on traffic in local areas of the net.

A slow response from a host with relevant documents can involve very heavy changes in the queue of URLs to be visited. Important documents can be lost, and the crawler will choose another "road" on the net. Another problem is related to the difficulty to judge the relevance of the crawl. As discussed in [14] it would be good to judge the relevance of the crawl by human inspection, even though it is subjective and inconsistent. However, this is not possible for the thousands of documents retrieved by the crawler.

Table 1: Starting points for collections

Collection	Core	Filter Components		
		3 WC	2 WC	Term
Algorithms	95	33	100	100
Programming Languages	118	20	20	100

Table 2: Threshold values used in tests

Collection	VSM	Term	2 WC	3 WC
Algorithms	0.18	0.5	0.1	0.08
Programming Languages	0.25	0.5	0.1	0.08

Our approach was used to compile large collections of Algorithms and Programming Languages. Note, these topics are close to each other. Table 1 shows the information concerning starting points for our collections. The column *Core* indicates the number of URLs from the starting seed relevant to the topic. Columns *3 WC* and *2 WC* include numbers of three and two word collocations manually selected by expert.

Table 2 presents threshold values used in runs of the crawler. The threshold for the *Dynamic Passage Partition* metric was selected which was two times smaller then the threshold for the *VSM* metric. Table 3 presents examples of three and two word collocations used in the filter. Word collocations consist of stems of words. Stems were produced by Porter's stemming algorithm.

Table 4 shows some statistics collected during the crawler run. In this table *Eval. docs* and *Recom. docs* indicate *evaluated documents* and *recommended documents* respectively. Documents, which were estimated as relevant using at last two metrics were put in the column *relevant undoubtedly*. The column *relevant heuristic* is needed for inclusion documents which were defined as relevant using the aforementioned heuristic. *VSM* and *DPP* indicate *Vector Space Model* and *Dynamic Passage Partition* metrics respectively. Columns from *non relevant* item include numbers manually detected non relevant documents.

The analysis of the amount of documents categorized as undoubtedly relevant was conducted. Precision was calculated using a standard formula. The results are very promising (see Table 5). Only 4% of documents from the collection of Algorithms and 2% from the collection of Programming Languages were appraised as non relevant to topics of interest. Non relevant documents are about laws, biographical information on scientists, mathematical education problems, etc. These types of documents were difficult to categorize accurately when defined manually.

Table 3: Examples of word collocations

Algorithms		Programming Languages	
3 WC	2 WC	3 WC	2 WC
unicod_collat_elem	minimum_distanc	object_orient_program	program_languag
travel_salesman_problem	closest_charact	assembly_languag_program	assembly_languag
binari_search_tree	samp_class	program_languag_design	abstract_class
abstract_state_machin	data_structur	abstract_data_type	sourc_code
approxim_string_match	genet_algorithm	visual_basic_programm	orient_program
theoret_comput_scienc	memet_algorithm	function_program_languag	compress_postscript
data_encrypt_standard	train_algorithm	abstract_class_definit	oper_system
train_algorithm_implement	data_compress	interc_program_languag	visual_basic
string_match_algorithm	pattern_match	structur_program_languag	data_type
spring_embedd_algorithm	data_type	sql_standard_process	common_lisp

Table 4: Statistics on collections

Collection	Eval. docs	Recom. docs	Relevant				Non relevant		
			undoubtedly	VSM	DPP	heuristic	VSM	DPP	heuristic
Algorithms	52180	8155	2678	626	2025	2826	113	402	598
Programming Languages	58215	8087	2980	1212	1008	2887	98	101	306

Table 5: Content of the undoubtedly relevant sets

Collection	Total docs	Rel. docs	Prec.
Algorithms	2678	2571	0.96
Programming Languages	2980	2923	0.98

Table 6: Content of the collections

Collection	Total docs	Rel. docs	Prec.
Algorithms	8155	7042	0.86
Programming Languages	8087	7582	0.93

The analysis of the whole content of our collections was also carried out. Results are shown in Table 6. A number recommended documents by the crawler are provided in the column *Total docs*.

It is extremely difficult to measure or even define recall for a domain–specific crawler because we have a rather incomplete and subjective notion of what is "good coverage" on a topic [15]. In any case, we attempted to calculate "recall". It should be noted: our estimation is very "rough" and subjective. But we think some measurement is preferred over no measurement. Estimated recall is presented in Table 7. To judge the recall of the crawl, we made selectively human inspection of filtered URLs. Documents filtered during the $10th$, $50th$ and $100th$ hours of the crawler run were downloaded except documents automatically determined as relevant. They were kept in the directories corresponding to their hosts. 20% of documents from each directory were appraised. If more that 75% of them were relevant to the topic of interest, then we assumed that all documents in the directory were also relevant except for manually detected non

Table 7: Estimation of recall

Collection	Total docs	Rel. docs	Recall
Algorithms	1419	520	0.51
Programming Languages	442	177	0.4

relevant ones. In the other case, we left only relevant documents from the evaluated set; all other documents were deleted. Remaining documents combined with documents recommended by crawler to give us an amount of "pseudo-relevant" documents (see the column *Total docs*). *Rel. docs* indicates a number of "pure" relevant documents after human inspection.

We have appraised the intersection of our collections. Cores do not include any shared URL, but they contain the following common hosts: *java.sum.com*, *www.cs.cmu.edu*, *ourworld.compuserve.com*, *personal.gip.fi* and *www.ccs.neu.edu*. Common elements of the persistent part of the filter are as follows. The terms are *code*, *string*, *implement*, *data*, *point*, *charact*, *pointer*, *design*, *optim*, *elem* and *dynam*. Two word collocations are *data_type*, *data_struct*, *script_languag*, *comput_system* and *formal_method*. The shared three word collocation is only one: *abstract_state_machin*.

The number of shared documents in both collections, which were automatically recommended, is satisfyingly small: It equals only 79 (less than 1% for our collections).

Statistics of filtering were accumulated. Measurements were made after every 2500 downloading of documents to be filtered. The number of recommended documents is between 255 and 840.

Table 8 provides the information regarding the number of

Table 8: Documents analyzed by hand

Collection	need to be analyzed by hand	Hand analysis		
		analysed by hand	non relevant	relevant
Algorithms	5477	1404	1113	4364
Program. Languages	5107	1061	505	4602

documents analyzed by hand. Numbers in the column *need to be analyzed by hand* are a difference between numbers taken from the column *Recom. docs* and *Relevant undoubtedly* (see Table 4). *Relevant* indicates documents from the set, *need to be analysed* documents, detected as relevant. Only about 17% of documents from the collection of Algorithms and 13% from the collection of Programming Languages were inspected by a human expert.

As we can see, all obtained results are very promising.

6. CONCLUSION

We have demonstrated a new method for topical resource discovery. We found, that using several metrics to estimate a relevance of documents is a powerful tool toward improving the quality of compiling thematic collections of text documents. Results have shown that our technique can help in the creation of domain–specific search engines. Presently the crawling is not so fast for the following reason: Since our major aim was the quality of the collections, the crawler ran with only one thread; operations to extract compressed files and to convert texts from PS and PDF formats into plain text are time–consuming; an ordinary connection to the Internet was provided for our experiments. The system can be used in practice without human intervertion to include documents recommended undoubtedly (relevant using at last two metrics) into a constructed collection. However in this case, many documents can be lost. Our aim in future work is to change the technology and to replace the human expert by a machine expert. In this case, the system may become more practical.

Collections constructed during our research have been incorporated into the OASIS system and they have been used by students at the University of Aizu (http://oasisntc.u-aizu.ac.jp/oasis).

7. REFERENCES

[1] A. Patel, L. Petrosjan and W. Rosenstiel, editors. *OASIS: Distributed Search System in the Internet.* St. Petersburg State University Published Press, St. Petersburg, Russia, 1999. (ISBN 5-7997-0138-0).

[2] Andrei Z. Broder, Steven C. Glassman and Mark S. Manasse. Syntactic clustering of the web. In *Proceedings of 6-th International World Wide Web Conference (WWW-6)*, 1997.

[3] Andrew McCallum, Kamal Nigam, Jason Rennie and Kristie Seymore. Building domain–specific search engines with machine learning technique. In *Proceedings of AAAI Spring Symposium on Intelligents Engine in Cyberspace*, 1999.

[4] R. Baeza-Yates and B. R. Neto. *Modern Information Retrieval.* Addison Wesley Longman Limited, England, 1999. (ISBN 0-201-39829-X).

[5] A. Birrell and P. Mc.Jones. The pstotext program. http://www.research. compag.com/ SRC/ virtualpaper/ pstotext.html.

[6] Charles Clarke, Antony Cox and Susan Sim. Searching program source code with a structured text retrieval system. In *Proceedings of SIGIR'99*, pages 307–308, Berkeley, CA, USA, 1999.

[7] D. A. Grossman and O. Frieder. *Information Retrieval: algorithms and heuristics.* Kluwer Academic Publishers, Norwell, Massachusetts, 2000. (ISBN 0-7923-8271-4).

[8] Harald Reiterer, Gabriella Mussler, Thimas M. Mann and Siegfried Handschuh. Insyder – an information assistant for business intelligence. In *Proceedings of SIGIR'2000*, pages 112–119, Athens, Greece, 2000.

[9] Keiichiro Hoashi, Kazunori Matsumoto, Naomi Inoue and Kazuo Hashimoto. Document filtering method using non–relevant information profile. In *Proceedings of SIGIR'2000*, pages 176–183, Athens, Greece, 2000.

[10] V. Kluev and V. Dobrynin. Crawling the web for topic–specific documents. In *Proceedings of International Conference on Performance Evaluation: Theory, Techniques and Applications*, pages 1–4, Japan, September 2000. The University of Aizu.

[11] C. D. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing.* The MIT Press, Massachusetts, 2000. (ISBN 0-262-13360-1).

[12] Mauro Marinilli, Alessandro Micarelli and Filippo Sciarrone. A case-based approach to adaptive information filtering for the www. In *Proceedings of Second Workshop on Adaptive Systems and User Modeling on the World Wide Web*, Canada, 1999.

[13] S. Mukherjea. Wtms: A system for collecting and analyzing topic–specific web information. In *Proceedings of 9th International World Wide Web Conference. The Web: 4 Next Generation*, 2000.

[14] S. Mascassy, A. Banerjee, B. Davidson, and H. Hirsh. Human performance on clustering web pages: A performance study. In *Knowledge Discovery and Data Mining*, volume 4, pages 264–268. 1998.

[15] Soumen Chakrabarti, Martin van den Berg and Byron Dom. Focused crawling: A new approach to topic-specific web resource discovery. In *Proceedings of Eight International World Wide Web Conference (WWW-8)*, 1999.

[16] V. Kluev, V. Dobrynin and S. Garnaev. Intelligent construction of thematic collections. In N. Mastorakis, editor, *Recent Advances in Applied and Theoretical Mathematics*, pages 103–106. World Scientific and Engineering Society Press, Athens, Greece, 2000. (ISBN: 960-8052-21-1).