Generating and Retrieving Text Segments for Focused Access to Scientific Documents

Caterina Caracciolo Maarten de Rijke

ISLA, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{caterina, mdr}@science.uva.nl

Abstract. When presented with a retrieved document, users of a search engine are usually left with the task of pinning down the relevant information inside the document. Often this is done by a time-consuming combination of skimming, scrolling and Ctrl+F. In the setting of a digital library for scientific literature the issue is especially urgent when dealing with reference works, such as surveys and handbooks, as these typically contain long documents. Our aim is to develop methods for providing a "go-read-here" type of retrieval functionality, which points the user to a segment where she can best start reading to find out about her topic of interest. We examine multiple query-independent ways of segmenting texts into coherent chunks that can be returned in response to a query. Most (experienced) authors use paragraph breaks to indicate topic shifts, thus providing us with one way of segmenting documents. We compare this structural method with semantic text segmentation methods, both with respect to topical focus and relevancy. Our experimental evidence is based on manually segmented scientific documents and a set of queries against this corpus. Structural segmentation based on contiguous blocks of relevant paragraphs is shown to be a viable solution for our intended application of providing "go-read-here" functionality.

1 Introduction

The growing number of scientific publications available in electronic format has changed the way people relate to documents. Working within the scientific domain, Tenopir and King [32] observe that researchers now tend to read more articles than before, but that, on average, the time dedicated to each article has shrunk and readers very rarely read an entire article—instead, they browse and skim the document, possibly doing attentive reading of only some parts of it. Increasingly, people use a "locate-and-read" strategy instead of the more traditional "read-and-locate" typical of a paper environment.

Currently, there are several examples where a kind of "go-read-here" functionality is available or being explored. For example, some general web search engines help users in their search "within" retrieved documents by providing links labeled "HTML version" (for non-HTML documents) and "In cache" (which takes the user to a cached version of the document where query words are highlighted). In the setting of document-centric XML retrieval, the search engine

looks inside the document for relevant information, and selects small relevant elements ("sub-documents") to be returned to the user [17].

Our setting is that of scientific literature digital libraries, and, more specifically, reference works such as surveys and handbooks in such libraries. Within this setting our aim is to provide "go-read-here" functionality of the following kind: given a query, suggest to the reader short, highly relevant segments from retrieved documents. How should we identify and retrieve appropriate segments for a "go-read-here" type of facility, using only query-independent information? Put differently, how should we create potential targets for hypertext links prior to knowing the link source (i.e., the query). Since every text has an internal structure [35], corresponding to the topics the author of the text wants to present, one obvious approach to identify the kind of segments we seek to identify is to adopt a so-called *structural* view on text segments, and take segments to be nothing but paragraphs. How does this strategy compare to so-called *semantic* segments, as produced by state-of-the-art segmentation algorithms such as TextTiling [13, 14] and C99 [6, 7]? These are the research questions that have guided much of the research on which we report in this paper.

Our main contributions are the following. First, we present an analysis of query independent text segmentation techniques applied to scientific texts. Second, we investigate the use of segments within a "go-read-here" retrieval task; in the process we define two new evaluation measures and also define a variation of precision to meet our needs. Our experimental evaluation is based on the *Handbook of Logic and Language* [34], a collection of 20 essays on the interface between logic and linguistics; each chapter (65 pages long, on average) is written by a different author and with varying internal organization and writing style. Our main finding is that structural segmentation based on contiguous blocks of relevant paragraphs is a simple but viable solution for our intended application of providing "go-read-here" functionality.

The rest of the paper is structured as follows. In Section 2 we present related work on "within document navigation." In Section 3 we survey relevant aspects of text segmentation methods. In Section 4 we describe experiments concerning document segmentation, and in Section 5 we present experiments concerning the use of these segments in a retrieval setting. We conclude in Section 6.

2 Related Work

Work related to this paper comes from research into hypertext link generation, information retrieval, information visualization, and digital libraries. The relations between two linked hypertext documents have been analyzed extensively [3, 9, 11, 33]. Information retrieval techniques have been used to generate hypertexts [1], and also text passages have played a role in generating links among documents [2]. In IR, passage retrieval refers to approaches that either return passages to the reader [27], or make use of evidence obtained from passages to improve the performance of full document retrieval [5, 18], or to select excerpts to index independently [14]; we follow the latter route.

Information visualization techniques have provided important means for improving the way in which documents are accessed, and especially the way in which *focused* retrieval is increasingly being facilitated. For example, TileBars [15] is a visualization paradigm for boolean retrieval, where documents are represented in their relative size, and passages within them are highlighted in color depending on the distribution of the query terms. SmartSkim [12] is a content-based browsing and skimming tool that divides a document deemed relevant by a user into fixed length sections represented by histograms whose height corresponds to the computed relevance of that section to a query.

In the context of digital libraries there has been considerable work on digitizing both content and metadata. Increasingly, methods are considered that integrate ideas building on traditional classification techniques developed over the centuries by librarians and information specialists with "free text" search technology, thus bringing in modern document retrieval technology [19].

Relatively little research has been aimed at providing focused access to *scientific* documents. Our work differs from the work carried out on generating hypertext in that we do not split the document into hyperlinked snippets, but, instead, provide the reader with a passage where she is to start reading, without missing out relevant text. In this sense, our work also differs from SmartSkim, in that we do not use fixed size passages. Finally, like TileBars, we presuppose that the document segmentation takes place offline, at indexing time, but unlike TileBars we aim at performing a comparison to understand which segmentation better suits the type of documents at hand.

3 Methods for Text Segmentation

Recall that our overall aim is to provide "go-read-here" functionality: return a highly relevant text segment from a long document in return to a user's query. Our first steps, then, will be to identify suitable text segments.

A segmentation is called *semantic* if segments are defined using a notion of the semantics of the text, *structural* if defined on the basis of structural information, e.g., paragraphs or sections, and *fixed size* if segments are identified through a fixed number of words or characters [30]. Many authors have proposed algorithms for semantic segmentation [16, 23, 25, 28, 29], either to achieve more accurate indexing [16] or to detect topic shifts in streams of news [31].

One of our core questions in this paper is to find out whether semantic methods offer an advantage over and above structural methods. Rather than implementing a new semantic segmentation method, or providing an exhaustive experimental comparison of all existing ones, we selected two well-known semantic methods for our experiments: *TextTiling* and *C99*. Both perform linear segmentation, the former based on cosine similarity between sliding windows of text, the latter based on divisive clustering. We chose TextTiling because of its established position in the literature (and because many other methods build on it); C99 was chosen because of the good results reported in the literature [6]. Below, we outline both segmentation methods; after that we compare the quality of the outputs of the two algorithms against the quality of structural methods

(Section 4), and examine the effectiveness of segments identified using either of the two methods for information access, again in comparison with structural methods (Section 5).

TextTiling [13, 14] tokenizes the document and performs stopword removal and morphological normalization. TextTiling divides texts into *pseudo-sentences* of a fixed length, which are grouped into pseudo-paragraphs, or *blocks*, of a fixed size, sliding along the text. Hearst [14] suggests that pseudo-sentences of twenty words and blocks of six pseudo-sentences work best in practice.

Each *gap* in between pseudo-sentences is assigned a cosine similarity value between pairs of adjacent blocks, computed with a sliding window mechanism. These values are then smoothed with a simple median smoothing algorithm [24] with a window of size 3, to eliminate small local minima, and the smoothed similarity values are then plotted against the sequence of gaps. The resulting plot is analyzed for peaks and valleys. Each gap is assigned a *depth score*, indicating how strong the evidence is that it is a candidate topic break. The depth score at gap g, $ds(g)$, is computed as $ds(g) = (a_s - g_s) + (b_s - g_s)$, where g_s is the smoothed similarity value at gap g and a_s and b_s are the smoothed similarity values at gaps a and b, to the left and to the right of g, respectively, each being a peak with respect to g. The deeper g is with respect to the closest valleys to the left and to the right, the more likely it is that the gap is a candidate break. Finally, TextTiling takes the gaps with the highest depth scores as candidate subtopic boundaries, but only places topic boundaries at (real) paragraph breaks.

C99 [6, 7] differs from TextTiling in that it takes real sentences as units and identifies topic boundaries by means of a divisive clustering method. First, the text is divided into tokenized sentences, then stop word removal and stemming follow. The algorithm then computes a similarity matrix at the sentence level, where the adopted similarity measure is the usual cosine similarity. Since the cosine measure is sensitive to the length of the sentences, Choi [6] applies a *ranking scheme* [22] to the similarity matrix to avoid using absolute values. Finally, a hierarchical divisive clustering method (based on [25]) is applied, where segment boundaries are selected to maximize a measure of internal segment cohesion. If the number of desired segments is not given up front, the clustering process is continued until no further segmentation is possible.

TextTiling has a clear intuitive interpretation in terms of text structure, while this is not the case for C99 (consider, e.g., the ranking scheme and the lack of references to specific textual or linguistic features). The experiments reported in [6] were performed on an artificially generated corpus of 700 samples, where a sample is a concatenation of ten text segments, where each segment consists of the first n lines extracted from a random document from the Brown Corpus.

4 Splitting Documents into Topic-Oriented Segments

Having introduced the two semantic text segmentation methods that we consider in this paper, our first aim is to see to how the segments they produce compare

	Chapter A	Chapter B		Chapter A	Chapter B
# pages	55	54	# segments	102	90
# section	13	3	# paragraphs/segm	1.6	2.5
# subsections	0	9	κ (inter-annotator agreement)	0.69	0.84
# paragraphs	168	223			
avg. par. length	458	320			

Table 1: (Left): Details about the corpus. (Right): Details about the ground truth for segmentation. Average paragraph length is given in number of words.

against a structural segmentation. Recall that structural segmentations in terms of paragraphs exploit the topic shifts (implicitly) marked by authors through their paragraph boundaries.

When applied to our data, consisting of long scientific documents, do TextTiling and C99 produce segments that are topically coherent? And: do they add anything when compared against two structural segmentations, into paragraphs and sections respectively? To answer these questions we developed a gold standard segmentation using two documents from our corpus, and used it to assess and compare the outputs of both the semantic and structural segmentation, as we will now describe.

4.1 Experimental Setting

First, a manually annotated corpus was created, containing "gold standard" topic breaks, to be used as the ground truth for evaluating the output of the structural and semantic segmentation algorithms. Two annotators independently annotated the text for topic breaks, and then discussed their results between them to come to a single annotation. The annotators were given basic guidelines:

1. a topic segment is a text snippet smaller than the original text and of homogeneous content;
2. segments do not overlap;
3. there are no topic breaks within paragraphs; and
4. no segment should span more than an entire section.

The corpus consists of two chapters—[36] and [21]—from the *Handbook of Logic and Language*, here called Chapter A and Chapter B, respectively (see Table 1, left-hand side, for details), with different internal structure and writing styles.[1] Chapters were in LaTeX format, which necessitated some preprocessing.

The right-hand side of Table 1 contains details about the annotators' output. The inter-annotator agreement, κ [8], indicates tentative reliability for Chapter A and high reliability for Chapter B (third row, right-hand side). The low κ score for Chapter A is probably due to the presence of long lists of examples and properties. This caused the annotators to have different perceptions about where an appropriate break between segments could be placed. The annotators agreed on a rather fragmented segmentation in case of Chapter A and on an only slightly more aggregative annotation in case of Chapter B.

[1] We counted as paragraph blocks of text separated by indentation, independently of the non-textual elements they can include (e.g., figures, tables, equations, ...).

	Chapter A				Chapter B			
	P	R	F	# Segm.	P	R	F	# Segm.
Paragraphs	0.61	**1**	0.76	168	0.41	**1**	0.58	223
Sections	**1**	0.10	0.18	13	**1**	0.02	0.04	3
TT default	0.62	**1**	**0.77**	165	0.42	0.98	**0.59**	212
TT s5-w20	0.61	**1**	0.76	169	0.42	**0.99**	**0.59**	215
TT s5-w30	0.61	**1**	0.76	166	0.42	**0.99**	**0.59**	215
TT s20-w30	**0.64**	0.83	0.72	132	**0.46**	0.79	0.58	157
TT s20-w40	**0.64**	0.80	0.71	128	0.43	0.71	0.54	150
C99 default	0.57	0.08	0.14	14	0.57	0.14	0.22	24
C99 r9	0.54	0.07	0.12	13	**0.62**	0.11	0.19	17
C99 r57	**0.72**	**0.13**	0.22	18	0.60	0.16	0.25	25

Table 2: Results for the two structural segmentations, and the best performing versions of TextTiling and C99. The highest values are in boldface.

4.2 Evaluation

We compared the segmentations produced by TextTiling and C99 with two structural segmentations: one in which each paragraph is a segment, and one in which each section is a segment. We used the implementations of TextTiling and C99 made available by Choi. We exhaustively explored the parameter settings for TextTiling (the number of smoothing cycles s, default = 5, and the window size w, default = 30 words) and for C99 (the size of the rank mask, default = 11×11, and the number of segments to find). Table 2 reports the results obtained with default values and with the best performing parameter settings. We report on precision (P), recall (R), and F-scores; P and R were computed on segment breaks, as opposed to entire segments.[2]

As expected, segments that are one paragraph long score best in recall but much less in precision, while sections do the opposite. C99 and the segmentation based on sections produce a similar number of segments and recall figures for Chapter A. In the case of Chapter B, they both score very low. This suggests that the quality of a segmentation is strictly related to the number and size of segments in the reference annotation.

C99 performs worst. When default parameters are used, the algorithm returns a few very long segments, too long to be of use in our intended focused retrieval application; varying the rank mask size does not yield significant change in the resulting segmentation. The stopping criterion used by the algorithm seems unsuitable to the type of text we deal with, and the good results achieved by C99 in the experiments reported in [6] do not carry over to our corpus.

TextTiling performs better on Chapter A than on Chapter B, and for C99 it is the other way around. This is related to the type of text and the type of segmentation they perform: TextTiling is more like a splitter (which matches with the Chapter A gold standard), while C99 is more like a lumper (matching

[2] In this way we look at how many segment boundaries are correctly identified, and we obtain a slightly more forgiving measure, with respect to counting how many entire segments are correctly identified.

with the gold standard for Chapter B).[3] The precision of C99 improves greatly when using a large rank mask (57) in the case of Chapter A, although recall remains very low.

We set out to find out whether TextTiling and C99 produce segments that are topically coherent, and whether they add anything when compared against two structural segmentations, into paragraphs and sections, respectively. The segments produced by C99 do not seem to be usable, given their low F-score. TextTiling and paragraph-based structural segmentation are on a par, both producing segments with reasonable F-score for one chapter (A) and mediocre F-score on another (B).

5 Retrieving Segments

Now that we have examined different ways of generating topically coherent segments from long scientific documents, our next aim is to use these segments in a retrieval setting. If we return relevant segments to users' queries, do we obtain segments that are "on target?" Do we obtain segments that are both relevant and a good starting points for reading? Are semantic segments better than structural segments?

To address these questions, we asked a single annotator (different from the two that created the gold standard segmentation described in Section 4) to create topics and mark up paragraphs in Chapters A and B for relevancy with respect to these topics. A baseline retrieval system was used to return ranked lists of segments for each of the segmentation methods and parameter settings listed in Table 2), and the outcomes were compared against the gold standard relevancy annotation. Below, we provide details about the development of the gold standard, the evaluation measures used, and the outcomes of our experiments.

5.1 The Gold Standard

We created a manually annotated corpus based on the same two chapters used in Section 4. A new annotator (different from the ones used for the gold standard creation in the previous section) developed a set of 37 queries and marked paragraphs in both Chapter A and Chapter B with respect to relevancy to each of the queries. The annotator was told to think of the annotation task in the following terms: you are helping to create a hypertext environment, with (possibly multiple) links from your topics into the corpus; you have to identify good link targets for your topics. The annotator was given the following constraints:

1. targets are non-empty sets of paragraphs;
2. the minimal target, i.e., the minimal unit of relevancy, is a single paragraph;

[3] The distinction between 'lumpers' and 'splitters' is used in lexicography to distinguish different behaviors in building of dictionary definitions. Lumpers look at similarities and tend to provide fewer definitions, broad enough to cover several cases; splitters look at differences and tend to provide more specific definitions, each covering a smaller set of cases.

3. if there are cross-references within a paragraph, do not also mark the text the cross-reference refers to (the text will be accessed in a hyperlinked form).

The annotator was given the chapters with no indication of the segmentation(s) produced in Section 4. The annotation resulted in an average of 2.1 and 7.1 relevant paragraphs per query, for Chapter A and Chapter B, respectively. In Chapter A relevant paragraphs are grouped in a single "block" per query, while in Chapter B there are, on average, 2.3 segments per query.

5.2 Evaluation Measures

We are interested in obtaining segments that are both relevant and good starting points for reading. Our task is similar to INEX in that we need to assess the relevancy of a document excerpt with respect to a topic, but arguably in our setting exhaustivity is less important than specificity, nor do we have the problem of document overlapping. Since we compare segments of varying length against a corpus where the unit for relevance assessment is the paragraph, we base our evaluation measures on paragraphs. In view of these considerations, we developed three measures: C-presision to determine the relevancy of a retrieved segment, and early onset error (EoE) and late onset error (LoE) to capture appropriateness of the start of the segment with respect to the distribution of the relevancy in the document. While C-precision corresponds to the (binary) notion of specificity in INEX, the two error measures were loosely inspired by [10].

C-precision is the proportion of relevant paragraphs included in a segment.

Early onset Error (EoE) measures the *proportion of non-relevant paragraphs* before the first relevant paragraphs in the segment. For a paragraph P, let r_P denote its rank in the document order (i.e., 1 for the first paragraph in the document, 2 for the second, etc.); by extension, for a segment S, r_S denotes the rank of the first paragraph in S. Then, for a query q and a retrieved segment S, $EoE(S) = 1$ if there is no block R of relevant paragraphs for q that overlaps with S, and otherwise $EoE(S) = \min\{1, (r_R - r_S)/|S| : r_R \geq r_S$ and R is relevant to q and overlaps with $S\}$, where $|S|$ is its size in number of paragraphs.

Late onset Error (LoE) measures the *proportion of missed relevant paragraphs* at the beginning of the segment. Using the same notation as in the definition of EoE, assuming that q is a query, and S is a retrieved segment, we define $LoE(S) = 1$ if there is no block R of relevant paragraphs that overlaps with S, and otherwise we put $LoE(S) = \min\{1, (r_S - r_R)/|R| : r_R \leq r_S$ and R is a relevant segment for q that overlaps with $S\}$.

A segment S with a perfect entry point, i.e., coinciding with the beginning of a relevant block R, will have $LoE(S) = EoE(S) = 0$.

A few quick remarks are in order. C-Precision depends on the size of the segment, as a segment consisting of only one relevant paragraph scores 1. The number of irrelevant paragraphs before the first relevant paragraph in a segment gives an indication of the effort required by the reader to reach relevant text. EoE has a bias for longer documents, since it divides the number of non-relevant paragraphs by the total number of paragraphs in the segment.

Segm. method	NoE[††]	Prop. NoE[†]	C-prec.[†]	Non-rel. segm.[††]	Non-rel. par. begin[†]	EoE[†]	Rel. par. missed[†]	LoE[†]
Paragraphs	**22**	**1.08**	**0.36**	**69**	**0.00 (0)**	**0.64**	2.11 (17)	0.71
Sections	13	0.83	0.07	78	7.77 (13)	0.72	3.00 (1)	0.67
TT default	**22**	1.06	0.35	70	0.00 (0)	0.65	2.12 (16)	0.71
TT s5-w20	21	1.00	0.33	72	0.00 (0)	0.67	2.05 (15)	0.73
TT s5-w30	21	1.03	0.34	71	0.00 (0)	0.66	2.12 (16)	0.72
TT s20-w30	**22**	1.06	0.32	71	1.00 (3)	0.67	2.05 (12)	0.70
TT s20-w40	**22**	1.06	0.31	71	3.50 (2)	0.67	**1.78 (12)**	0.69
C99 default	6	0.92	0.09	78	6.44 (16)	0.75	4.88 (5)	**0.66**
C99 r9	4	0.83	0.06	**81**	10.78 (19)	0.81	6.00 (1)	0.67
C99 r57	7	0.97	0.10	77	6.00 (17)	0.76	4.88 (5)	0.67

Table 3: Summary values for all algorithms considered, across all queries. Highest scores per measure are in boldface. ([†]) results averaged over all queries. ([††]) total number.

5.3 Evaluating the Retrieval of Segments

We will now evaluate the retrieval of segments, using the 37 topics developed. We use a basic retrieval engine based on the vector space model, with tf.idf term weighting and settings that are known to be beneficial for the retrieval of short documents [26]. In Table 3 we report on the following measures: total number of topics for which an exact entry point was returned (no onset error, NoE); average proportion of retrieved segments with no onset error, average C-precision; total number of non-relevant segments; average number of non-relevant paragraphs at the start of segments returned; average EoE; average number of relevant paragraphs missed at the start of segments; and average LoE.

The measures described above are applied at cut-off three; i.e., we capture the situation where we are returning three targets per query. Results are reported in Table 3, where columns 2–8 correspond to the measures listed above.

When a segment only contains one non-relevant paragraph, the entire segment can only be counted as non-relevant, Similarly, the longer a segment is, the more likely it is that it also contains non-relevant paragraphs, possibly placed at the beginning of the segment. The number of non-relevant segments retrieved when segments are as long as entire sections suggests that tf.idf tends to discriminate short documents better than long ones.

As in the previous section, the results for TextTiling are similar to those of the single paragraph structural segmentation. This is due to the length of the segments, which is similar in the two cases. Analogously, when C99 is used, the retrieval algorithm finds approximately as many non-relevant segments as in the case of segments one section long.

All C99 versions perform only slightly better than segmentation by sections. The single paragraph segmentation has lowest average EoE, a fact that is ex-

plained by the high precision: since a single paragraph can only be either totally relevant or totally irrelevant, it follows that in case of many relevant segments, there will be many zeros in the average. This is also witnessed by the fact that C-precision and EoE sum to one for this system. C99 with default settings scores the highest EoE, due to the large size of the segments. Concerning the LoE, this time the lower error rate is scored by C99 with default parameters, immediately followed by the baseline based on paragraphs.

Discussion. The experiments on which we reported in this section were aimed at investigating the use of structural vs. semantic text segmentation as a basis for providing "go-read-here" functionality. Structural segmentation (in terms of single paragraphs) scores best according to many of the measures adopted: in some cases this is due to the length of the segments (e.g., C-precision and EoE), in other cases it is due to the sparsity of the relevant text in the reference corpus. The fact that LoE is higher for the single paragraph structural segmentation and TextTiling suggests that, in case of documents with more dense relevancy, it is useful to retrieve longer segments than just paragraphs. This issue could also be addressed by aggregating paragraphs after the retrieval phase, which will also help in case of documents with sparse relevancy with respect to the query. In order to address this issue, it could be good to aggregate paragraphs after the retrieval phase, and only then form the segment to return to the user.

6 Conclusions and Future Work

In this paper we reported on an analysis of query-independent text segmentation methods aimed at supporting "go-read-here" type of functionality in the setting of scientific literature digital libraries. We focused on two aspects: generating segments and retrieving segments as focused responses to user queries. For both aspects we had to develop ground truth data and evaluation measures.

For the generation of segments our main finding was that the presence of formulas, tables and long list of examples, together with the presence of different kinds of internal references, made the annotators divide the documents in a very fragmented way, which resulted in very competitive scores for the structural segmentation into paragraphs. As to retrieving segments, we found that the structural segmentation into paragraphs is hard to beat using the semantic segmentation methods that we considered. We conjecture that it may be beneficial to aggregate paragraphs after retrieving them.

Now, a number of caveats apply. First, we only worked with one corpus, albeit with chapters authored by different people. It remains to be seen to what extent our findings generalize to other corpora. Second, we treated the issue of search within a document as a passage retrieval task, where we assume that the passages are independent: relevancy of one paragraph does not imply relevancy of earlier or later paragraphs. It would be interesting to see whether a more sophisticated model that captures dependencies between paragraphs improves the retrieval scores. Third, we assumed that segmentation can be done off-line, independent of user queries. We have not investigated whether text segments

can best be established with respect to the question one intends to ask, in which case it is worthwhile integrating the segmentation and the retrieval phases so that segments can be defined on the basis of the query posed.

For future work, it is interesting to see to which extent some level of discourse semantics might be used to improve both the identification of segments and their retrieval. Along similar lines, we suggest looking in more detail at cue phrases, even though there is mixed evidence in the literature (they have been shown to be misleading in [4] but useful in [20]).

Acknowledgments This research was supported by Elsevier and by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 365-20-005, 612.000.106, 612.000.207, 612.-013.001, 612.066.302, 612.069.006, 640.001.501, and 640.002.501.

References

[1] M. Agosti and J. Allan, editors. *Methods and Tools for the Automatic Construction of Hypertext*. Elsevier Science Ltd, March 1997. Special Issue of *Information Processing and Management* Volume 33.

[2] J. Allan. Building hypertext using information retrieval. *Information Precessing and Management*, 33(2):145–159, March 1997.

[3] L. Baron, J. Tague-Sutcliffe, M. T. Kinnucan, and T. Carey. Labeled, typed links as cues when reading hypertext documents. *Journal of the American Society for Information Science*, 47(12):896–908, 1996.

[4] G. Brown and G. Yule. *Cambridge Textbooks in Linguistics Series*. Cambridge University Press, 1983.

[5] J. P. Callan. Passage-level evidence in document retrieval. In *Proc. of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland*, pages 302–310, July 1994.

[6] F. Choi. Advances in independent linear text segmentation. In *Proc. of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL-00)*, pages 26–33, 2000.

[7] F. Choi. Linear text segmentation: approaches, advances and applications. In *Proc. of CLUK3*, 2000.

[8] J. Cohen. The coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 21(1):37–46, 1960.

[9] J. Conklin. Hypertext: An introduction and survey. *Computer*, 20(9):17–41, 1987.

[10] A. P. de Vries, G. Kazai, and M. Lalmas. Tolerance to irrelevance: A user-effort oriented evaluation of retrieval systems without predefined retrieval unit. In *Recherche d'Informations Assistee par Ordinateur (RIAO 2004)*, April 2004.

[11] S. J. DeRose. Expanding the notion of links. In *Proc. of Hypertext'99*, pages 249–257, 1989.

[12] D. J. Harper, S. Coulthord, and S. Yixing. A language modeling approach to relevance profiling for document browsing. In *Proc. of JCDL*, 2002.

[13] M. A. Hearst. *Context and Structure in Automated Full-text Information Access*. PhD thesis, University of California at Berkeley, 1994.

[14] M. A. Hearst. Multi-paragraph segmentation of expository text. In *Proc. 32nd ACL*, 1994.

[15] M. A. Hearst. Tilebars: visualization of term distribution information in full text information access. In *Proc. of CHI'95*, 1995.
[16] M. A. Hearst and C. Plaunt. Subtopic structuring for full-lenght document access. In *Proc. of the 16th Annual International ACM SIGIR Conference on Research and Development in IR*, pages 59–68, 1993.
[17] INEX. INitiative for the Evaluation of XML Retrieval, 2004. http://inex.is.informatik.uni-duisburg.de:2004/.
[18] M. Kaszkiel and J. Zobel. Passage retrieval revisited. In *Proc. of SIGIR 97*, pages 178–185, 1997.
[19] M. Lesk. *Understanding Digital Libraries*. The Morgan Kaufmann series in multimedia information and systems. Morgan Kaufmann, second edition, 2005.
[20] C. Manning. Rethinking text segmentation models: An information extraction case study. Technical Report SULTRY-98-07-01, University of Sydney, 1998.
[21] R. Muskens, J. van Benthem, and A. Visser. Dynamics. In *Handbook of Logic and Language*. Elsevier, 1997.
[22] M. O'Neill and M. Denos. Practical approach to the stereo matching of urban imagery. *Image and Vision Computing*, 10(2):89–98, March 1992.
[23] J. M. Ponte and W. B. Croft. Text segmentation by topic. In *European Conference on Digital Libraries*, pages 113–125, 1997.
[24] L. W. Rabiner and R. W. Schafer. *Digital processing of speech signals*. Prentice-Hall, Inc., 1978.
[25] J. C. Reynar. *Topic Segmentation: Algorithms and Applications*. PhD thesis, University of Pennsylvania, 1998.
[26] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 25:513–523, 1988.
[27] G. Salton, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In *Proc. of the 16th Annual International ACM/SIGIR Conference, Pittsburgh, USA*, pages 49–58, 1993.
[28] G. Salton, J. Allan, and A. Singhal. Automatic text decomposition and structuring. *Information Processing and Management*, 32(2):127–138, 1996.
[29] G. Salton, A. Singhal, C. Buckley, and M. Mitra. Automatic text decomposition using text segments and text themes. In *Proc. of the 7th ACM Conference on Hypertext, Washington D.C., USA*, 1996.
[30] E. Skorochod'ko. Adaptive method of automatic abstracting and indexing. *Information Processing*, 71:1179–1182, 1972.
[31] N. Stokes, J. Carthy, and A. F. Smeaton. Segmenting broadcast news streams using lexical chaining. In T. Vidal and P. Liberatore, editors, *Proc. of STAIRS 2002*, volume 1, pages 145–154. IOS Press, 2002.
[32] C. Tenopir and D. W. King. Reading behaviour and electronic journals. *Learned Publishing*, 15(4):159–165, October 2002.
[33] R. Trigg. *A network approach to text handling for the online scientifc community*. PhD thesis, University of Maryland, 1983.
[34] J. van Benthem and A. ter Meulen, editors. *Handbook of Logic and Language*. Elsevier, 1997.
[35] T. van Dijk. *Some Aspects of Text Grammar*. Mouton, 1972.
[36] J. van Eijck and H. Kamp. Representing discourse in context. In *Handbook of Logic and Language*. Elsevier, 1997.