Topic driven access to scientific handbooks

Caracciolo, C.

CHAPTER 5

Looking for Link Targets

Data is not information,
which is not knowledge,
which is not wisdom.
(Bruce Horn)

Deep inside... it's shallow.
[Cooper, 1981]

After the presentation of a map for the domain of logic and linguistics (Chapter 3), we performed user studies to assess its usability (Chapter 4). The user studies confirm that the hierarchical structure is intuitive and the interface usable. In particular, they confirm that users find it important to have associated with each topic in the map both larger text fragments and concise glosses. In this and the following chapter, we explore the issue of *linking* the map to the handbook so as to identify the "larger text fragments" that we just mentioned. The issue of linking resources to one another has been widely addressed in the hypertext community, where numerous attempts have been made to clarify the "meaning" of a link, typically through classifications or taxonomies of link types. In the present chapter we do not attempt to propose a new categorization or new categories of links. Instead, we concentrate on the automatic identification of *link targets*, i.e., text fragments to which the link points. The selected links will be links of "aboutness," where the notion of aboutness is established by means of a *similarity measure*, as used in Information Retrieval (IR), between a representation of the topic in the map and the text. (For a survey on the notion of "aboutness" in subject indexing see the classical paper [Hutchins, 1977].)

In our perspective a link target is a textual document, or rather an excerpt or segment of a text, such that:

1. it includes only relevant content, with respect to a given topic, and

2. it is "self-contained" from the point of view of readability.

In other words, according to the first constraint above, the segment should be maximally specific, while the second constraint involves what we call rules of "politeness" to the reader: for example, anaphora should be resolved within the text, and sentences with adverbs suggesting inferences (like "so," "then," "therefore," and so on) should not be separate from their premises. Also, it is important that the segment include relevant content and that the relevant part be right at the beginning of it.

Since every text has internal structure, corresponding to the topics the author of the text wants to present, one obvious approach to get at 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 segmentation* produced by state-of-the-art segmentation algorithms that are based on features of the text and aim at reflecting its content?

In Information Retrieval, passage retrieval techniques have been widely studied to improve the result of the document search task [Kaszkiel and Zobel, 1997, Liu and Croft, 2002, Salton et al., 1993b, Wilkinson, 1994]. Approaches to passage retrieval can be classified on the basis of the unit they consider as a passage. *Structural passage retrieval* takes as units the very same unit into which most texts are already divided (sections, subsections, etc.). Another passage retrieval technique is based on *fixed-sized windows*, i.e., sliding windows of text, whose size is usually expressed in terms of the number of words they contain. Finally, it is possible to look for more semantically oriented segments: if this is the case, the process of identifying passages is often called *topic segmentation*.

We explore and compare two structural passage retrieval methods (when units are single paragraphs and when they are entire sections) and two well-known algorithms for topic segmentation: TextTiling [Hearst, 1997] and C99 [Choi, 2000a,b]. We apply these segmentation methods to the *Handbook of Logic and Language*, and in order to evaluate the results, we build a gold standard, i.e., a reference corpus, by manually annotating *topic breaks* in two out of the twenty chapters in the handbook.

In the next chapter we address the connection between segments and the topics in the map. We then report on a second set of experiments, where each segment is considered as an individual "document," or rather pseudo-document, that can be linked to. All together, these segments form a collection of pseudo-documents to retrieve using queries derived from the topics in the map. The retrieved segments become then our candidate link targets. The evaluation of these targets is done on the basis of a second gold standard, that we created manually by annotating the relevance of paragraphs in the handbook with respect to the query.

The remainder of this chapter is organized as follows. In Section 5.1 we present the main approaches to passage retrieval: structural, fixed size, and semantic (often called topic segmentation). In Section 5.2 we describe two algorithms for linear topic

segmentation: TextTiling and C99. In Section 5.3, we discuss the evaluation of algorithms for topic segmentation and describe the creation of an annotated corpus for that purpose. In Section 5.4 we present our first set of experiments, which are designed to establish how good the considered topic segmentation algorithms are at identifying topic segments in a scientific domain. In Section 5.5 we discuss the results we obtained, and in Section 5.6 we draw conclusions.

5.1 Passage Retrieval

Information retrieval deals with the retrieval of documents in answer to a query expressing a user's information need. In the case of long documents, the estimated relevance to a query can be "diluted" by the size of the document. For this reason IR researchers have looked at the possibility of analyzing *passages* of a document, either to gain *evidence* about the relevance of the whole document, or to retrieve passages instead of full documents. We use the expression "passage retrieval" to refer to IR approaches that look at passages for one of these purposes. The contribution of a passage to document retrieval performance, typically in terms of ranking, is often called *evidence*.

In the rest of this section we will illustrate the principles of the three approaches to passage retrieval, structural, fixed size and semantic, then we concentrate more in depth on the semantic passage retrieval.

Applications of passage retrieval in document retrieval include the following:

- reducing the amount of text to present to the user [Salton et al., 1993b],
- selecting excerpts to index independently [Hearst, 1994b],
- improving ranking [Callan, 1994, Kaszkiel and Zobel, 1997].

Reduction of the amount of text presented to the user is often applied after the document retrieval phase, as happens in automatic Question Answering (QA), to select the portion of text from which to extract the answer. When the focus is on indexing, passage retrieval helps to select homogeneous excerpts of text characterized by a high degree of internal coherence and a high degree of dissimilarity to the rest of the text. The resulting excerpts are indexed independently of the rest of the document. When used to improve ranking, similarity values are computed for all passages with respect to the query. Then the score of the document (determining its position in the ranking) is computed on the basis of a combination of the scores for the whole document and for the passages. In general, passage retrieval is used when a high degree of precision is required, as in QA, where it is particularly helpful in the case of long documents [Monz, 2003]. Passage retrieval also finds applications in text summarization [Mani and Maybury, 1999], where it is used to select homogeneous and relevant excerpts to summarize.

5.1.1 Structural Passage Retrieval

Usually, published textual documents are divided into parts, such as sections, subsections and paragraphs. In structural passage retrieval those parts are used as passages. If documents are encoded in a markup language (e.g., XML, SGML, or LaTeX), passages are naturally identified by some elements of the markup (e.g., sections or subsections).[1] Salton and Buckley [1991] propose a method to use the vector space similarity between segments of texts (e.g., paragraphs) in order to compute a finer notion of similarity between (full) documents. According to their proposal, two documents are similar if they exhibit both a *local* and a *global similarity*. Global similarity means looking at the similarity between the two documents (or between a document and a query), while local similarity means looking at the maximum value of pairwise similarity between the components of the document. Salton and Buckley [1991] normalize the term frequency component in order to make the measure independent of paragraph length. Later, Allan [1995] uses the notion of local/global similarity for the automatic generation of typed links among full documents. Wilkinson [1994] investigates the contribution of structural passage evidence to document retrieval, where passages are sections. His conclusion is that passage evidence can help document retrieval, and in particular that the benefit from using passage evidence is most obvious when (high) precision is more important than recall.

Hearst [1994a] argues that markup information, such as the division of text into paragraphs, is less reliable for segmentation than information extracted from the content. According to Hearst, this is a reason to chose semantic passages. In the case of high quality publications such as the *Handbook of Logic and Language*, the internal division of the text into paragraphs tends to be reliable in terms of internal homogeneity. What changes across the book is the average size of the paragraphs and the internal division of the chapters, both related to the writing style of the authors and to the internal organization of the text. In particular, the fact that chapters can be organized with different levels of granularity (cf. Table 5.2) makes it difficult to decide beforehand what unit to use. Also, sometimes a paragraph can be too short to exhaustively include a topic, while entire sections can be too long.

5.1.2 Fixed Size Passage Retrieval

Paragraphs, and sections for that matter, may vary much in length, thus affecting passage retrieval. One way to overcome this problem is to use length normalization procedures, like Salton and Buckley [1991]. Another way is to consider a passage as consisting of a fixed number of words (or characters, or sentences). In the following we investigate this approach.

[1] Since 2001, INEX, the INitiative for the Evaluation of XML retrieval [Fuhr et al., 2006], has studied the potential of using XML document structure for retrieval purposes, considering every element to be a retrievable unit. For reasons explained below, we do not want to go "below" the level of paragraph, and will view paragraphs as our "atomic units."

Callan [1994] studied the contribution made by fixed size passage retrieval to document retrieval. A passage consists of a window of 100, 200, or 300 words (the value is fixed empirically, depending on the collection), evaluated at query evaluation time instead of during a preprocessing phase. The window slides along the text, overlapping by half of its length (so that each window, except the first and the last ones, overlaps two others): this is the reason why it is called a *sliding window* technique. The starting point is the location of the first identified occurrence of a query term in the document. Callan compares three different strategies for document retrieval: (i) using the similarity between the query and the entire document, (i) using the similarity between the query and the best passage, and (iii) using a combination of the first two strategies (i.e., again a combination of local/global similarity). His results suggest that the latter method yields the best results. Note that length normalization is not a problem for this approach, but semantic ambiguity is. The sliding window approach is widely used, for example in QA, where the passage length is often expressed in terms of natural language sentences [Lee et al., 2001, Light et al., 2001, Llopis and Vicedo, 2001]. As we will see in the next section, a semantically oriented passage retrieval algorithm such as TextTiling also uses the sliding window technique as an intermediate step to identify semantically oriented passages.

The fixed size approach has proven to be useful in both document retrieval and question answering for computing evidence from passages. It has also shown to be more effective (in terms of contribution to document retrieval) than approaches to passage retrieval based on structural passages [Kaszkiel and Zobel, 1997]. Unfortunately, fixed size passage retrieval provides no indication of where to split the document in a reader-friendly manner, therefore it is not suitable for the definition of candidate link targets. In this respect, structural passage retrieval and semantically oriented passage retrieval are better options to overcome that problem.

5.1.3 Semantic Passage Retrieval

Neither of the two approaches to passage retrieval presented so far, structural and fixed size, *explicitly* addresses the issue of dealing with the "content" of the passage. Obviously, the fixed-size approach does not consider content, but even in the structural approach content is not emphasized. In fact, while a paragraph can be too short to capture a "topic" (which can span two or more paragraphs, for example, the first being devoted to a definition, the second to an explanation of the definition with examples or comments), an entire section can be too long. Semantic passage retrieval attempts to define passages on the basis of the semantics of the texts, i.e., on the basis of the topic in the text. For this reason it is also known in the literature as *topic segmentation*.

Hearst and Plaunt [1993] propose to identify semantic passages for the purpose of more accurate indexing. Other approaches have centered on the possibility of recognizing topic shifts in streams of news [Stokes et al., 2002]. From our point of view, the possibility of correctly identifying a topic means that we can present the reader with a readable excerpt from the text, are sensibly shorter than the entire document. In our

view, a readable text should not include unresolved anaphoric references, nor should discourse connectives such as "on the one hand, [...] on the other hand" be disconnected. In general, salient argumentation blocks should be in the same segment. The anaphora phenomenon is the most widely studied: Morton [1999] finds that looking back two sentences makes the antecedent available 98.7% of the time, while looking back only one sentence makes the antecedent available 96.9% of the time. He does not report data about the availability of the antecedent with respect to their collocation in the paragraph, but his observation suggests that both reference and antecedent tend to be placed in the same paragraph. In order to take into consideration the antecedent, we would need a discourse parser (work on this is currently ongoing, see for example [Webber, 2004]). All these considerations encourage us, or rather force us, to take paragraphs as an atomic unit for topic segmentation (which then becomes semantics-based paragraph aggregation).

Among the many proposed algorithms for topic segmentation [Hearst and Plaunt, 1993, Ponte and Croft, 1997, Reynar, 1998, Salton et al., 1996a,b], we have selected TextTiling and C99. We chose TextTiling because of its established position in the literature, and C99 because of the good results reported in [Choi, 2000a]. In the next section we outline two families of approaches to topic segmentation, linear and hierarchical, and describe the two linear algorithms chosen in detail.

5.2 Linear Topic Segmentation: Two Algorithms

A topic segmentation is *linear* if the identified segments follow one another, with no distinction between topic and subtopic, and no possibility of nesting subtopics to form topics. In contrast, a segmentation is *hierarchical* if it allows nesting of topic segments. For example, Grosz and Sidner [1986] and Mann and Thompson [1987] adopt a hierarchical model of discourse, also embodied in the approach to topic segmentation taken in [Yaari, 1997].

A discussion on whether hierarchical segmentation better describes the argumentative structure of a text than a linear one is beyond the scope of this thesis. For our purposes, though, a linear segmentation is preferable because it allows us to concentrate on the shift from topic to topic, without keeping track of the topic/subtopic organization of the text. In other words, it is an appropriate tool to delimit excerpts from the text to connect to the map.

In the following we describe in detail TextTiling and C99.

5.2.1 TextTiling

TextTiling [Hearst, 1994a,b] is probably the best known algorithm for linear topic segmentation of expository texts. Following Skorochod'ko [1972], Hearst assumes a linear structure of text and looks at word occurrences in order to trace the development of a topic.

The algorithm tokenizes the document (i.e., divides the text into individual lexical units), and preprocesses it in order to minimize the variety of forms of the same word (i.e., it applies stop word removal, and stemming or morphological normalization of the remaining words). TextTiling divides texts into *pseudo-sentences* of fixed length, which are grouped into pseudo-paragraphs, or *blocks*, of fixed size sliding along the text. The length of the pseudo-sentence and the size of the blocks are the two adjustable parameters of the algorithm. Hearst found that pseudo-sentences of twenty words and blocks of six pseudo-sentences work best in practice. Real paragraphs are not taken as the unit for computing similarity, so as to avoid a length normalization step.

The sliding window method described above allows one to compute a cosine similarity value between all pairs of adjacent blocks. Each pseudo-sentence gap is assigned a similarity value resulting from the comparison of the blocks forming a window interfacing at that gap. Figure 5.1 gives a schematic representation of a text divided into blocks of equal size. Windows sliding along the text are represented below the line: the first window starts from the beginning of the text and spans six sentences (from gap 0 to gap 6), the algorithm computes the similarity value between its adjacent window (from gap 6 to gap 12), and assigns it to gap 6 (the little circle between the two windows). The next window starts at gap 1, the following one at gap 2 and so on.

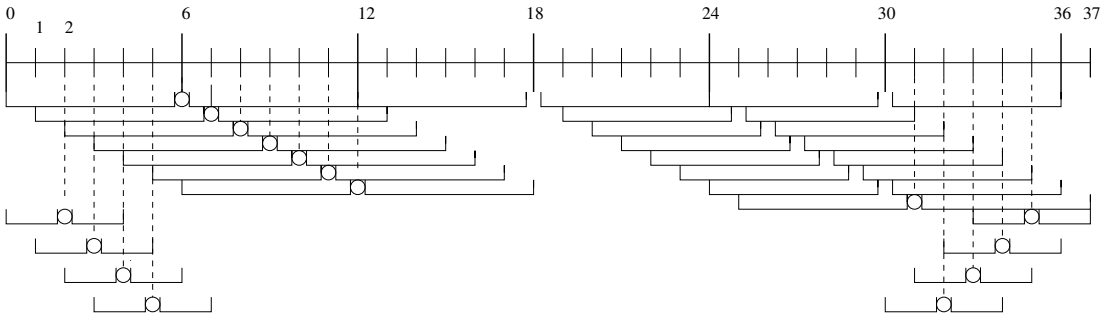

Figure 5.1: Grouping of blocks by sliding windows in TextTiling.

Gaps before 6 and those at the end of the text obtain a similarity value by using smaller windows, of size 2 blocks.

Similarity values are not directly used to compute the paragraph breaks, because they only indicate the grade of similarity between adjacent windows of text, without indication of how that value of similarity has to be interpreted in the picture of the distribution of similarity along the text. Therefore, TextTiling uses a more sophisticated procedure to compute topic breaks. After assigning a similarity value to each gap, these values are smoothed by using a discrete convolution. Next, the result is further smoothed with a simple median smoothing algorithm [Rabiner and Schafer, 1978] with a window of size 3, to eliminate small local minima. The smoothed similarity values are then plotted against the sequence of gaps, and the resulting plot is analyzed for peaks and valleys. Each gap is assigned a *depth score*, indicating how strong the evidence is that the present gap is a candidate topic break. Let $ds(g)$ be the depth

score at gap g, it is computed as $ds(g) = (a_s - g_s) + (b_s - g_s)$, where 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. In other words, the value assigned at each gap is computed by summing the difference between the similarity value at g and the peak to the left and the difference between the similarity value at g and the peak to the right. The intuition is that the deeper g is with respect to the closest peak 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 highest depth score as candidate subtopic boundaries, but only places the topic boundary at a (real) paragraph break, after checking that breaks do not fall too close to one another.

Summarizing, TextTiling has a clear intuitive interpretation in terms of text structure and solves the problem of varying paragraph length by using a combination of sliding window and rounding to the original paragraph breaks.

5.2.2 C99

C99, the algorithm introduced by Choi [2000a,b], 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 the preprocessing phase (stop word removal, stemming) follows. 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 applies a *ranking scheme* (based on the one proposed in [O'Neill and Denos, 1992]) to the similarity matrix to avoid using absolute values. The *rank mask* is the square defining the neighbor region around a certain value in the matrix: Choi suggests using a 11×11 mask. Each value in the similarity matrix is then replaced by the number of neighboring elements with lower similarity value, i.e., each similarity value is replaced by its rank. Moreover, since the rank is incomplete along the borders of the matrix, the values have to be normalized somehow: Choi proposes to use the proportion of elements with a lower value over the total number of elements examined. It follows that when using an 11×11 rank mask, the range of the new value is between 0 and 10. Finally, a hierarchical divisive clustering method (based on [Reynar, 1998]) is applied. At each step in the clustering procedure, a segment boundary is selected such that it maximizes a measure of internal segment cohesion. If the number of desired segments is not given, the clustering process is continued until no further segmentation is possible. For all segmentations, the algorithm computes a measure of internal cohesion of the clustering, and the change in the internal cohesion from one segmentation to the next. Then, the algorithm chooses as optimal segmentation the one whose change in internal cohesion is above a certain threshold, computed from the change in internal cohesion of the entire sequence of segmentations.

C99 performs topic segmentation without explicitly referring to specific textual or linguistic figures. It performs a linear segmentation, where the linearity is implicitly obtained by the degree of similarity between close and distant sentences in the text.

Also, the application of the ranking scheme has no straightforward interpretation in linguistic terms, nor does it have a mechanism to stop the clustering procedure. The experiments reported in [Choi, 2000a] were performed on an artificially generated corpus of 700 samples. A sample was taken to be a concatenation of ten text segments, where each segment consists of the first n lines extracted from a randomly chosen document from the Brown Corpus.[2] The performance of the algorithm was evaluated in terms of efficiency (the average number of CPU seconds required to process a test sample) and in terms of effectiveness: its precision was evaluated using the P_D measure we describe in Section 5.4.1. We decided to use C99 in our experiments in order to see whether the good performance shown in those experiments could be replicated when using a "real life" text collection instead of an artificial one.

5.3 Building the Ground Truth

A text segmentation algorithm can be assessed in three ways: by comparing it against a manually annotated corpus (annotated by the author of the texts, or by other annotators), by comparing it against an "automatically" annotated corpus, or extrinsically, by assessing how the segmentation improves the computational task in which it is used.

Since the authors of the texts are usually not available for annotation, it is common practice to elicit the annotation from other people, who may be given little or no instruction about how the segmentation should be performed. If this is the case, the aim of the annotation is to capture the intuitive notion that people have about what a topic is. For example, Hearst [1994b] and Klavans et al. [1998] do not supply precise guidelines to the people that perform topic boundary annotations, in order to see if and what similarities emerge from their annotation. A common way to automatically generate an annotated corpus is to take a corpus of news-wires (with annotated topic breaks), eliminate breaks between news, and possibly mix the news at random. The goal of the topic segmentation system in that case is to rediscover the original topic breaks. This is the type of evaluation used in the experiments described in [Choi, 2000a].

We decided to build an annotated corpus by annotating two chapters from the *Handbook of Logic and Language* and use them as a gold standard. Our purpose is to experiment with the difficulties that a real corpus offers, and have a chance to reflect on the peculiarities of working with a scientific handbook. The expected advantage of this approach (as opposed, say, to using an automatically created corpus) is that we have a model of how segmentation for our purposes should be done. The disadvantage is that the process is time consuming and will lead to a relatively small corpus. In the following we describe and discuss the creation of a corpus of manually segmented topics. In Section 5.3.3 we discuss the topic boundary annotations of the *Handbook of*

[2] The Brown Corpus of Standard American English was the first of the modern, computer readable, general corpora. It was compiled by W.N. Francis and H. Kucera from Brown University (USA). The corpus consists of one million words of American English texts printed in 1961. The texts for the corpus were sampled from 15 different text categories to make the corpus a good standard reference.

Logic and Language.

5.3.1 Inter-Annotator Agreement

As with many other annotation tasks, the style and quality of the annotation depend on the specific annotators. This is a well-known issue in the area of both information retrieval and computational linguistics [Lesk and Salton, 1969].

Many annotators, many annotations Since any annotation depends on the annotator who performs it, it can be useful to recall the distinction between 'lumpers' and 'splitters,' as it is defined in lexicography to distinguish different behaviors in the 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. The same distinction is also used by Klavans et al. [1998], who asked a group of naïve readers (i.e., with no previous notion of topic segment nor instructions about how to perform the task) to annotate a text for topic boundaries. The authors found that the differences among the performed annotations could be described in terms of splitter or lumper annotators, since a few important topic breaks would be recognized by all people, while other less prominent topic shifts would be perceived as a topic break only by some.

On a more formal level, Passonneau and Litman [1993] discuss at length the issue of agreement among annotators in a discourse segmentation task.[3] The authors ask seven subjects to segment a number of discourse transcripts using an informal notion of speaker intention; when four or more agree on a boundary, the boundary is considered significant using a variation of the Q-test [Cochran, 1950]. In practice, this methodology considers the majority as equivalent to the judgment of an experienced reader, where judgments are expressed on the basis of an intuitive notion of what a topic is. This approach is used by Hearst [1994b], although the author notes that it is not clear how useful the data about the significance is, because the simple majority can be insufficient to objectively claim the presence of a topic boundary. Perhaps a better measure of inter annotator-agreement is given by the *Kappa* test [Cohen, 1960], that also takes into account the agreement by chance as dependent on the number of categories the annotators have to mark—note that in case of segment boundaries there are exactly two categories: break/no break. Here is the definition of $Kappa$:

$$Kappa = \frac{P(A) - P_e}{1 - P_e}, \tag{5.1}$$

where $P(A)$ is the fraction of times that annotators agree, and P_e represents the agreement by chance. The value of *Kappa* decreases when P_e increases, i.e., when there

[3] That study aimed at finding out whether an intuitive notion of speaker *intention* is sufficiently understood by naïve subjects to yield significant agreement on segment boundaries in corpora of oral narratives.

	Chapter *A*	Chapter *B*
# pages	55	54
# sections	13	3
# subsections	0	9
# subsubsec.	0	21
# paragraphs	168	223
avg. par. length	458	320

Table 5.1: Details about the corpus.

is a high degree of agreement by chance. The agreement by chance, P_e, between two annotators is computed as follows:

$$P_e = \sum_{j=1}^{n} \left(\frac{\frac{C_j}{2}}{N} \right)^2 = \frac{1}{4N^2} \sum_{j=1}^{N} C_j^2 \tag{5.2}$$

where j represents the number of categories available (in this case two: break vs. no break), C_j is the sum of the numbers of paragraph breaks that have been assigned the same label j by both annotators. Finally, N is the total number of paragraph breaks.

5.3.2 The Handbook of Logic and Language

The *Handbook of Logic and Language* consists of 20 chapters, on average about 65 pages long, each written by different authors and on independent, though related, subjects. The style of writing varies from chapter to chapter, with different usage of anaphora, different internal structure, and different use of formulas, tables, equations, examples and cross references. However, all chapters have a typical linear writing style, with the assumption that the reader would read the entire chapter, if not the entire handbook, from beginning to end. Also, cue phrases[4] are used with varying frequency: for example, in some more narrative chapters each paragraph might begin with phrases like "As we mentioned above..." or "Therefore," while others make less use of these.

We chose two chapters from the *Handbook of Logic and Language*, that we call here Chapter *A* and Chapter *B* to build our annotated corpus.[5] Although similar in length (on paper, approximately 55 pages each), the two chapters exhibit different internal structure and also different writing styles. Table 5.1 summarizes some quantitative aspects of the two chapters.

[4] A cue phrase is a phrase used to highlight specific moment in a text or speech. Examples of cue phrases include: "As we have previously seen...," and "As previously discussed...." Cue phrases in other domains include: "This was X reporting from Y," "Ladies and gentlemen...," "The next speaker is ...".

[5] We used Chapter 3 [van Eijck and Kamp, 1997] and Chapter 10 [Muskens et al., 1997] of the Handbook.

We remark that the notion of paragraph in this domain turned out to be rather flexible, mainly because of the many non-textual blocks inserted in the text, such as equations, derivations, definitions, lemmas, propositions and theorems, figures, tables and so on. For example, one would assume that paragraphs are marked and recognizable by indentation, and that equations or list of examples should not force indentation, i.e., a new paragraph. The extent to which this style is applied varies throughout the handbook and in the two selected chapters. We counted as paragraph blocks of text separated by indentation, independently of the non-textual elements they can include (e.g., figures, tables, equations,...). Finally, when counting the length in paragraphs we have only considered words longer than two letters (to discard in-line formulas), and have not considered the bibliographic items when counting the length in pages. As we counted these figures by processing the LaTeX sources, which were written according to different styles, some noise is likely.

Chapter *A* is organized into 13 sections, with no other internal structuring than the paragraph division and blocks of definitions, propositions and so on, explicitly marked as such. Occasionally, paragraphs are characterized by a title, in some cases, they are as long as half a page (about 458 words), in a few other cases they are as short as two sentences, usually when they only consist of definitions, propositions or axioms.

Chapter *B* consists of 3 sections organized into respectively 0, 4 and 5 subsections, spanning 54 pages in the printed version; the text is distributed into 223 paragraphs. Chapter *B* does not contain examples and theorems distinguished as such, nor tables and only a few figures, but it does contain many in-line formulas and lists of formulas (axioms or properties). As in Chapter *A*, paragraphs in Chapter *B* can be quite long, up to the entire length of a the smallest unit (subsection), and on average 320 words long.

5.3.3 The Resulting Ground Truth

We asked two annotators to independently annotate the texts for topic breaks, and then agree on a single annotation. The annotators were given basic guidelines that did do not define a strict notion of topic segment. To the best of our knowledge, no strict guidelines for topic segmentation of written texts are available, despite the fact that there are guidelines available in related efforts such as discourse segmentation [Nakatani et al., 1995]. The recommendations given to the annotators were:

1. a topic segment is an excerpt of text smaller than the original text and of homogeneous topic;
2. segments do not overlap;
3. no segment breaks should be placed within paragraphs;
4. no segment should span more than one section;

The manual annotation of Chapter *A* results in 102 segments, each on average 1.6 paragraphs long, while the manual annotation of Chapter *B* results in 90 segments,

	Chapter *A*	Chapter *B*
# segments	102	90
# paragraphs/segments	1.6	2.5
Kappa (inter-annotator agreement)	0.69	0.84

Table 5.2: Details about the ground truth for segmentation.

on average about 2.5 paragraphs long. The inter-annotator agreement computed by *Kappa* is 0.69 for Chapter *A* (tentative reliability) and 0.84 for Chapter *B* (high reliability). When deciding on the final annotation, the two annotators did not make radical changes, as is reflected by the values of *Kappa* between each annotation and the final version (on average 0.81). All figures are summarized in Table 5.2.

The fact that the *Kappa* score for *A* is only 0.69 is due to the presence of long lists of examples and properties discussed at length in the text. This caused the annotators to have different perceptions about where an appropriate break between segments could be placed, although they placed approximately the same number of breaks (103 and 104). One difficulty in annotating the chapters was the varying frequency of cue phrases used, especially at the beginning of paragraphs, and used to link one paragraph to the preceding one. Another difficulty stemmed from the presence of cross references to examples or equations previously defined in the text. Chapter *B* is a good example of the former case, since the authors often smooth the transition between paragraphs by using phrases like: "Again, ..." or "As previously noted..." This problem was somewhat compensated for by the higher degree of internal structure of the text, compared to Chapter *A*, so that the overall level of inter-annotator agreement is high. The *Kappa* score between the final annotations and the original organization in sections and subsections gives a measure of how much more detailed the final annotation is with respect to the structure of the text: we have a very low value for Chapter *A* (0.39), the chapter with more shallow structure, and a higher level (0.69) for Chapter *B*, with a deeper structure.

The use of cue phrases is related to the publication medium. In fact, the use of references at the beginning of paragraphs is meant to smooth the connection between paragraphs when reading on paper, but it is arguably less necessary when reading on screen, where more nimble constructions (such as hyperlinks) can be adopted. The presence of cross references should be considered when giving recommendations to the annotators. For example, recommending that cross references between pieces of text far apart should not be connected: they can be connected later by hyperlinks.[6] Topic segmentation is especially useful in cases like Chapter *A* that exhibit a shallow text structure and with long lists of examples discussed through the text. The second issue is stylistic, strongly related to the publication medium.

[6]Such a recommendation was actually given to annotators in the next set of experiments (see Section **??**).

One difficulty we did not take into account when devising the segmentation is the difficulty of distinguishing between different levels of presupposition. It turned out to be not always clear when a reference to something previously introduced could be safely split and when the splitting would prevent the reader from having the *background* necessary to understand the text.

5.4 Evaluating Link Targets Against the Ground Truth

We compared the segmentation performed 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 could only use sections because this is the only internal subdivision common to both chapters). We used the implementation of TextTiling and C99 provided by F. Choi and freely available from his web page.[7] All segmentation methods were applied to the two annotated chapters (Chapter *A* and *B*) from the *Handbook of Logic and Language*.

The adjustable parameters for the TextTiling implementation we used are the number of smoothing cycles (default = 5) and the window size (default = 30 words). The adjustable parameters for C99 are the size of the rank mask and the number of segments to find. The default rank mask is 11×11, while if no number of segments is set, the algorithm will use the stop criterion described in Section 5.2.2.

5.4.1 Evaluation Measures for Topic Segmentation

Assuming we have a reference corpus with annotated segment boundaries, we need a metric to measure how good or bad our segmentation algorithm is compared to the standard.

The measures most commonly applied for tasks such as ours are the standard *precision* and *recall*, applied either to topic breaks or to entire segments, and the P_D *precision measure* [Beeferman et al., 1997]. When considering precision and recall as applied to topic breaks we look at how good the system is at recognizing a topic shift. By applying the measure to entire segments we look at the ability of the system to recognize homogeneity within the text. The results will vary accordingly, as the following example depicts. Let $T = \{c,d,e,f\}$ be a fragment of text, where each letter stands for a paragraph, and let $Ref = \{(c,d,e,f)\}$ be the manual annotation for it (meaning that c, d, e, f are grouped into the same segment). Let $TS1$ and $TS2$ be two alternative segmentations for T: $TS1 = \{A\}$ and $TS2 = \{B,C\}$ (Figure 5.2). Now, if we look at entire segments, $TS1$ has returned a correct segment, while $TS2$ has not; looking at topic breaks, $TS1$ has 2 correct topic breaks out of 2, while $TS2$ has two correct out of 3. Although crude measures (they do not give a measure of how distant the hypothesized segment break is from the real break), precision and recall are well understood.

[7] URL: http://www.cs.man.ac.uk/~mary/choif/software.html. Code downloaded in January 2004.

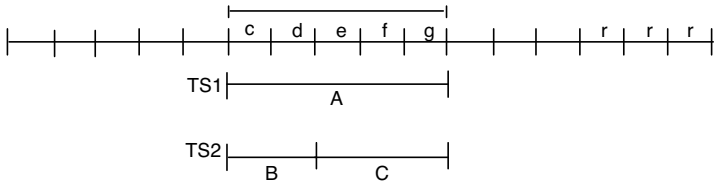

Figure 5.2: Difference between looking at precision and recall of entire segments vs. topic breaks.

Arguably, there can be various "degrees" of error, also depending on the user model adopted. For example, missing a topic break by only one paragraph is not as serious an error as missing it by ten paragraphs, and one can argue that a topic break placed too early is a more acceptable mistake than a topic break placed too late. For this reason, Reynar [1998] suggests judging a boundary correct if it appears within a fixed-sized window of words around an actual boundary. The disadvantage of this measure is that it does not distinguish between correct and incorrect boundaries within the window.

Beeferman et al. [1999] describe a probabilistic error measure P_D for document segmentation: $P_D(r,h)$ is the error of a hypothesized segmentation h with respect to a reference segmentation r. The P_D error measure is defined as follows. Let $I = \{1,2,\ldots,n\}$ be a set indexing the units (paragraphs or sentences) of a document T. For each $i, j \in I$, let r_i denote the segment to which unit i is assigned in some reference segmentation r, and h_i similarly denote the assignment of unit i in a hypothetical segmentation h. The error measure, then, is:

$$P_D(r,h) = \frac{1}{2} \sum_{i,j \in I} D(|i-j|)(\delta_{r_i}(r_j) \bar{\oplus} \delta_{h_i}(h_j)), \tag{5.3}$$

where δ_x is the Kronecker delta function:

$$\delta_x(y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise,} \end{cases}$$

the $\bar{\oplus}$ operator is the boolean XNOR operation:

$$x \bar{\oplus} y = \delta_x(y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise,} \end{cases}$$

and D is a pre-defined probability distribution defined over all possible distances between units in T. One can interpret P_D as rewarding hypothetical segmentations that agree with the reference by grouping related units and separating unrelated ones. If the two segmentations agree on every pair of units, $P_D(r,h) = 1$, while if they disagree on every pair, P_D will be very low ($P_D(r,h) = 0$ only if D is chosen such that $D(0) = 0$).

The choice of the probability distribution D is central to the behavior of the error measure P_D. If D is chosen to be uniform, P_D is simply the probability that any two

Segmentation method	Chapter *A*			Chapter *B*		
	Precision	Recall	# Segm.	Precision	Recall	# Segm.
Paragraphs	0.61	**1**	168	0.41	**1**	223
Sections	**1**	0.10	13	**1**	0.02	3
TextTiling	0.62	**1**	165	0.42	0.98	212
C99	0.57	0.08	14	0.57	0.14	24

Table 5.3: Values of precision and recall on topic breaks for the four segmentation methods. TextTiling and C99 are applied with all parameters set to their default values. Best scores are in **boldface.**

randomly chosen units are correctly identified as belonging to the same segment or not. This choice for D tends to be too forgiving, however, as it gives equal weight to distant, and hence likely unrelated, units that even the most simplistic segmentation algorithm is liable to correctly separate. The authors instead suggest using a distribution that concentrates the probability mass on a small range of distances, or even on a single, empirically derived distance.

The disadvantage of the P_D precision measure is that it depends on the length of the document, in the sense that in case of a non-trivial segmentation, it is likely that two distant units are not hypothesized as belonging to the same segments—i.e., provided that the whole document is not in a single segment. Since our documents are indeed rather long, we decided not to use the P_D measure. We adopted instead the precision and recall measures applied to topic breaks, because in this way a system producing an output that agrees with the annotation for topic breaks is rewarded even if the size of the segment is not correct.

5.4.2 Results

Table 5.3 shows the performance of the two structural segmentation methods, C99 and TextTiling with default parameters. The evaluation measures adopted are precision and recall computed on segment breaks; the highest values are in **boldface**.

When assuming that segments coincide with paragraphs, all possible breaks are obviously found, which leads to the maximum recall value for both Chapter *A* and Chapter *B*. Under the same assumption, precision is higher for Chapter *A* than for Chapter *B*, which has to do with the fact that reference segments for Chapter *B* are longer than Chapter *A*.

When assuming that segments coincide with sections, we have maximum precision for both chapters, because all section breaks are also topic breaks in the annotation. In contrast, recall is very low for Chapter *A* and even lower for Chapter *B*, which is a direct consequence of the few segment breaks hypothesized.

TextTiling returns almost as many segments as the number of paragraphs, which ensures a recall close (Chapter *B*) or identical (Chapter *A*) to the highest possible. Also, the values of precision follow the same pattern as in the case of segments one paragraph long, although slightly higher.

TextTiling	Chapter *A*			Chapter *B*		
	Precision	Recall	# Segm.	Precision	Recall	# Segm.
default	0.62	**1**	165	0.42	0.98	212
s5-w20	0.61	**1**	169	0.42	**0.99**	215
s5-w30	0.61	**1**	166	0.42	**0.99**	215
s5-w40	0.61	0.89	163	0.42	0.96	209
s10-w30	0.62	0.96	159	0.43	0.95	200
s10-w40	0.61	0.94	157	0.44	0.91	190
s20-w30	**0.64**	0.83	132	**0.46**	0.79	157
s20-w40	**0.64**	0.80	128	0.43	0.71	150

Table 5.4: Tuning parameters for TextTiling. Best scores are in **boldface**

C99	Chapter *A*			Chapter *B*		
	Precision	Recall	# Segm.	Precision	Recall	# Segm.
default	0.57	0.08	14	0.57	0.14	24
r7	0.53	0.08	15	0.61	0.15	24
r9	0.54	0.07	13	**0.62**	0.11	17
r11	0.57	0.08	14	0.61	0.15	24
r13	0.54	0.07	13	0.57	0.14	24
r15	0.57	0.08	13	0.57	0.14	24
r17	0.57	0.08	14	0.57	0.14	24
r57	**0.72**	**0.13**	18	0.60	0.16	25

Table 5.5: Tuning parameters for C99. Best scores are in **boldface**

Finally, C99 has the lowest precision scores for Chapter *A*, but the highest for Chapter *B*. On the other hand, recall falls dramatically for Chapter *A* and is very low for Chapter *B* (but higher than in the case of "sections as segments"). The low value of recall is related to the low number of segment breaks hypothesized, and only about half of those agree with the gold standard.

Next, we experiment with tuning parameters for TextTiling (results are shown in Table 5.4). We tried different combinations of smoothing cycles (*s* in the table) and window size (*w* in the table). By enlarging the window size from 30 to 40 words, but keeping the same number of smoothing cycles, we find little variation in both precision, recall and number of hypothesized segments. On the other hand, an increase in the number of smoothing cycles has a clearer effect, especially on the size of the segment. By increasing the number of smoothing cycles the relative differences among similarity values are smoothed and the resulting segments will be longer. As a consequence, fewer segments breaks are hypothesized.

Table 5.5 shows differences in performance when varying the size of the rank mask (r in the table) for C99. The algorithm with default parameters results in a very ag-

	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 5.6: Summarizing the results for the two structural segmentations, and the best performing versions of TextTiling and C99.

gregative segmentation, and results change little when changing the parameters. The rank mask size does not affect the result of the segmentation in terms of number of segments hypothesized. We did not use the other parameter available, the number of segments in which to divide the text, because for our purposes a topic segmentation algorithm should not be told in advance how many segments there should be (and how long). Only for the sake of comparison with TextTiling, though, we imposed the same number of segments as hypothesized by the best versions of TextTiling. As Table 5.5 shows, both precision and recall increase, together with the number of hypothesized segments.

5.5 Discussion

The parameter variation in TextTiling and C99 did not yield a dramatic performance improvement (cf. Tables 5.3, 5.4 and 5.5). In particular, C99 turns out to be the worst performing algorithm: when default parameters are used, the algorithm returns few, very long segments, too long to be of effective use in this application. Varying parameters does not yield significant differences in the resulting segmentation. We hypothesize that the reason for this is that the stopping criterion used by the algorithm is not suitable to the type of text we deal with. We deem it important that an algorithm for segmentation in this domain be able to decide by itself how many segments it should return. In other words, the algorithm should be able to detect topic shifts on the basis of parameters other than the exact number of segments to return. Moreover, the parameters in C99 are difficult to interpret from a linguistic point of view, which makes it hard to hypothesize about and interpret the effect of tuning them. We suppose that the good results the algorithm achieved in the experiments reported in [Choi, 2000a] are related

to the kind of collection it was evaluated against, and to the measure of accuracy used (the P_D measure we discussed in Section 5.3).

TextTiling behaves better on Chapter *A* than on Chapter *B*, while it is the other way around for C99. This is probably 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 with the Chapter *B* gold standard). The precision of C99 improves greatly using a large rank mask (57) in the case of Chapter *A*, although recall remains very low. TextTiling gives the best balance between precision and recall.

Table 5.6 summarizes the performance of the two structural segmentations and that of TextTiling and C99 with default parameters and with best results obtained. The two structural segmentations exhibit a somewhat symmetrical behavior: segments that are one paragraph long score best in recall but very low in precision, while segments one section long exactly the opposite. On the other hand, C99 and the segmentation based on sections produce a similar number of segments and similar recall for Chapter *A*. In the case of Chapter *B*, they again score very low and again there is a correlation between the number of segments and the recall value. This suggests that the quality of a segmentation is strictly related to the number and size of segments in the reference annotation.

5.6 Conclusions

In the course of this chapter we addressed our Research Question 3, which reads: "What are suitable targets in the handbook to establish focused, topic driven links from topics in our browsable map?" We assumed that passage retrieval techniques should serve our purpose and compared two structural segmentation (by paragraph and by section) techniques and two semantic passage retrieval techniques (TextTiling and C99). In order to perform a comparison, we created a ground truth by manually annotating two chapters of the *Handbook*, and compared the segments generated by the four automatic methods selected with the segments marked relevant by the annotators. As evaluation measure, we used precision and recall applied to topic breaks. We found that the best algorithms for this task are TextTiling and the structural segmentation by paragraphs, while C99 and the structural segmentation by sections are to be avoided. The tuning of parameters of TextTiling did not give sensible improvement of the results.

The results in Tables 5.3 and 5.5 indicate to what extent the considered algorithms are able to repeat the segmentation made by the human annotator. As previously noted, they do not say anything about how good the resulting segmentation is as a basis for the production of candidate links to be connected to the map. In other words, our Research Question 4 is still to be addressed. The work we present in the next chapter addresses that question. There, we report on our experiments with the retrieval of the four sets of segments described in this chapter. That is, in the next chapter we will consider

what happens when a collection of segments is considered as a collection of candidate link targets. For that purpose we will use both algorithms—TextTiling and C99—with default parameters, and the combination of parameters reported in Table 5.6.