Combining visual layout and lexical cohesion features for text segmentation

Min-Yen Kan

January 29, 2001

1 Abstract

We propose integrating features from lexical cohesion with elements from layout recognition to build a composite framework. We use supervised machine learning on this composite feature set to derive discourse structure on the topic level. We demonstrate a system based on this principle and use both an intrinsic evaluation as well as the task of genre classification to assess its performance.

2 Introduction

A *document structure tree*[1] can be defined as a data structure that allows navigation of a document by sections. These trees can be hierarchically organized, having subsections of sections and may embed special items, such as figures, tables or hyperlinks. They may be used directly by an end user for document access, or indirectly through other applications.

This paper describes a strategy to compute document structure using a framework that deals both with rich, semi-structured documents with layout features as well as impoverished, text stream-like documents. Our system, the **C**ombined **L**ayout **A**nd **S**egmentation **P**reprocessor (CLASP), performs

[1] or variously, *logical structure tree* (Summers, 1995), or *document map* (Zizi and Beaudouin-Lafon, 1995).

this task by computing visual layout and lexical cohesion features, and then combining them using supervised machine learning.

3 System Architecture

In previous systems for text segmentation (Kozima, 1993; Hearst, 1993; Beeferman et al., 1997), section breaks were determined solely by linguistic means, such as lexical cohesion. However, text today is often more than a stream of words. It is often presented to the reader with document access structures (e.g., headers and lists) and formatting to make the material easier to find; these layout features also contribute to segmentation. We believe that the combination of layout recognition and lexical cohesion techniques allows for a wider range of texts to be dealt with. The two streams of information are independent of each other, and can be represented by orthogonal sets of features. Our hypothesis is that in many texts these two streams would complement each other, reinforcing decisions on section breaks. With certain data sources, one of these streams may not be present (i.e. no layout information in some emails), and the system could then rely on the other to provide information.

Thus, one of the design goals of CLASP is to provide a framework to combine and balance these data. We use machine learning to decide what types of information are salient. Machine learning allows the system to be customized automatically for a particular domain, and allows for training over a general collection to establish reasonable default rules.

Since lexical cohesion works best on content-oriented structures[2] (e.g., prose) and presumably not as well on layout-oriented structures (e.g., section headers or captions), CLASP provides lexical analysis only for content-oriented prose text. Thus, the layout preprocessing module (`Layser`) makes a first-pass decision on whether lines of the text are layout- or content-oriented. Content-oriented lines and layout-oriented lines are analyzed separately by the lexical cohesion module (`Coheser`) and by the header analysis module (`Header`), respectively. The two streams of information are rejoined in the

[2]Content-oriented and layout-oriented structures are relative ends of a single scale, discussed in (Summers, 1995).

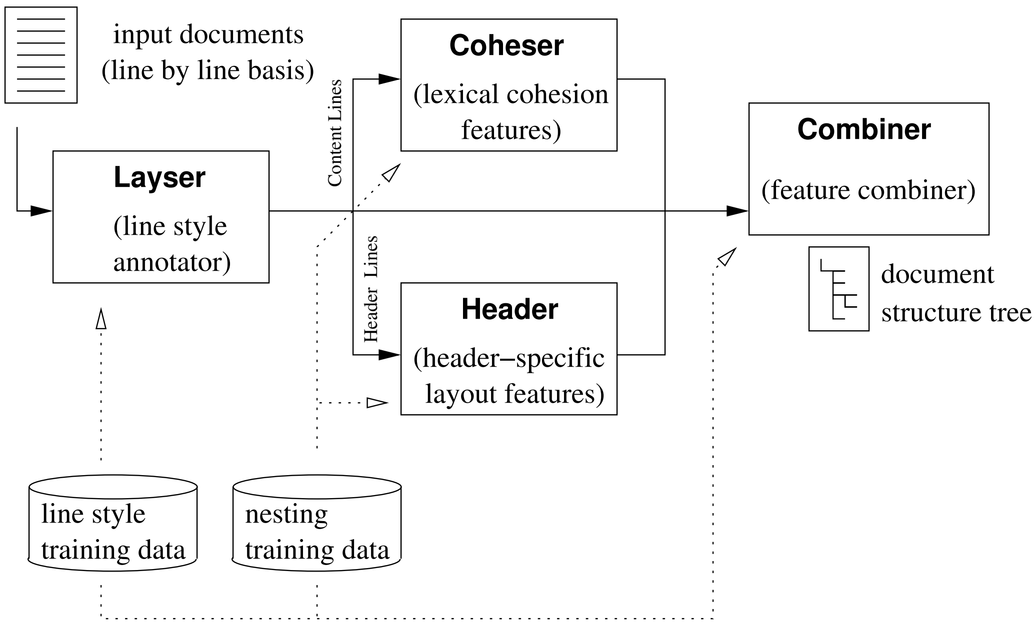

Figure 1: CLASP system architecture

final machine learning module (`Combiner`), which determines the document structure tree, as shown in Figure 1.

Another design goal in CLASP is wide applicability. For this reason, we use plain ASCII documents as input. Plain text is used as an original text form (e.g., in email) and as a lowest common denominator, and thus any type of textual data can be transformed into it. These transformations can be lossy (e.g., font sizes and weights are lost), and with richer original formats, CLASP's performance may not be optimal. However, modules can be built to provide additional features that represent the enriched formatting, as shown in Figure 2. Other researchers have enumerated additional factors that contribute to document structure in these enriched input formats (e.g., with document block information (Chen et al., 1999), or font information (Klein and Fankhauser, 1997)).

For simplicity, we choose the line as the unit of granularity. We chose lines instead of sentences or paragraphs because it is easily accessible, robust, and not dependent on linguistic processing.

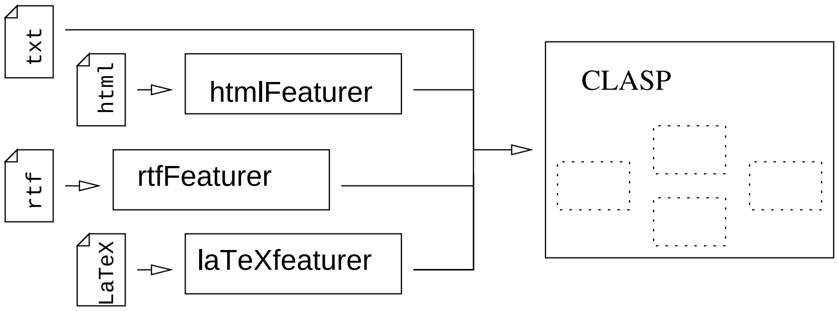

Figure 2: Extending CLASP system architecture to work with richer text formats.

We will now describe CLASP's modules in order of execution.

3.1 Line style annotator module (`Layser`)

The development of information retrieval methods that query structured data (Loeffen, 1994) has increased the importance of both understanding the effects of layout as well as generating it. In CLASP, layout properties that are available in plain text are translated into features by the `Layser` module. `Layser` outputs an application-driven logical style (explained later) for each non-blank line.

Input Features. `Layser`'s machine learning features represent the visual cues available in plain text documents. We categorize the features into five groups similar to (Esposito et al., 1994), all shown in Table 1 with example values:

1. **Spacing (4 features):** Both intra- and interline spacing are given as separate features. We also include alignment features such as the left and right margins, which are normalized against the widest values found in the document.

2. **Marking (1 feature):** Orthographic case (e.g., upper or lower case) is our sole marking category feature. Font family, size and weight would be included here when CLASP is extended to handle more enriched text formats. It is calculated as the average value of the individual words'

case.

3. **Punctuation (4 features):** These features capture different types of line-final punctuation (marking prose text) as well as line-initial ones (marking list items). To capture embedded headers (e.g., headers in-line with content), we also record the position of the first occurrence of a sentence-like punctuation.

4. **Word (1 feature):** Just the bag of words contained in the line. These can be used to find specific cue words that mark a particular logical style.

5. **Document (4 features):** These features address the overall characteristics of the document. We encode the approximate position of the line in the document, the document length, the document's average orthographic case, and the document's average number of interline blanks. This provides a method for passing exception information to the machine learner (e.g., when the entire document is in uppercase, it is essentially caseless and rules that use case should be dropped).

To model local context dependencies, `Layser` computes the first four feature categories for each line and for its neighboring previous and next non-blank lines. These $(4 + 1 + 4 + 1 * 3 =)$ 30 features are added to the 4 document features to make the final 34-dimensional feature vector used by `Layser`.

Output classes. `Layser` makes a first pass decision on the line's logical style, and categorizes the line as one of 11 styles in Table 2. The line styles we use are motivated from the viewpoint of prospective applications: what types of logical styles might applications like to treat differently? For example, (Luc et al., 1999) showed that the syntax and style of enumerated list items have semantic meaning (i.e. bulleted items versus ordered items) and should be processed with the understanding of these semantics. Similarly, page information lines should be discarded in content processing but can help determine meta information about the text.

Motivating the output categories from the perspective of a client application affects what we consider a logical style. Attribute/Value pairs are often regarded as a specific kind of list item, but in CLASP have been given their own logical style because they can be directly inserted into a database. Since

␣Input␣Features.␣␣Layser's␣machine␣learning␣features␣represent␣the␣visual␣cues␣available␣in

Name	Description	Sample Value
Spacing Features		
intraSpace	average space between words (* 100)	$\frac{13}{12} * 100 = 108$
interSpace	number of blank lines after	0
leftMargin	normalized to min left margin (widest: 0)	$\frac{1}{100} * 1000 = 10$
rightMargin	normalized to min right margin (widest: 1000)	$\frac{90}{100} * 1000 = 900$
Marking Features		
case	orthographic case (300 $\equiv$ all words uppercase)	$\frac{(3*2)+(9*1)}{12} * 100 = 125$
Punctuation Features		
initPunct	type of line initial punctuation	0
listNext	for lists; 1 if looks like next expected item	0
embedPunct	percentage position of first non-initial punctuation	$\frac{14}{89} * 1000 = 157$
endPunct	type of line final punctuation	0
Word Features		
words	bag of words in the line	"input" "features" ...

Table 1: Sample line and calculated `Layser` feature values. Document category features not shown.

styles are motivated from the perspective of a generic application, we exclude genre- and domain-specific logical styles, such as Bibliography or Salutation.

With our classification scheme, some lines could be classified as belonging to multiple classes: such as when List Item information is presented in Tables. To avoid this problem, we asked annotators to label each line's most salient logical style with respect to the end application. We made an exception in adding the Embedded Header tag, which marks a line as containing normal discourse text prefaced with an in-line header, since headers are needed to correctly infer segmentation structure.

At this point, `Layser` creates the features and then applies its machine learned model to produce a first pass classification of each line. Layout-oriented lines are passed to the `Header` module for further layout analysis, while content-oriented lines are passed to the `Coheser` module for lexical cohesion analysis.

Line Style Class	Abbreviated Description
Table	part of the body of a table
Separator	rulelines, lines to separate sections
Attribute/Value	left half of line has value in right half
List Item	(non-section like) list item
Embedded Header	headers on same line as content text
Header	headers, titles, subheaders
Caption	text attached to a picture, figure or table
Page Information	document openers, trailers, ToCs, page numbering
Auxillary Text	secondary content (not main: bibliography, abstract, sideboxes)
Main Content Text	Content-oriented prose text
Unknown	*default tag, used for error checking*

Table 2: The inventory of logical line styles in the `Layser` and `Combiner` modules.

3.2 Header priority module (`Header`)

This module receives Header and Embedded Header lines from `Layser`, and produces a feature set targeted at classifying the segment nesting depth of these lines. The initial feature set used in this task is similar to `Layser`: case, spacing, and punctuation features are all used. We add an additional feature, representing header scope, giving the percentage of the document that the header has immediate scope over (i.e. until the next line with a header line style).

Section headers may not manifest themselves in the same manner across documents; in one document the title may be centered, but in another it may be in boldface type. The *relative difference* between these features across headers within a document seems to dictate their nesting depth. `Header` thus computes its final feature set based on the differences in the values of these initial features in adjacent headers, shown in Table 3. This corresponds to learning whether one header dominates, is dominated by, or is on parity with an adjacent header. These pairwise features are `Header`'s output and are passed on to the `Combiner` final machine learning module.

Previous Header

`␣␣␣␣Line␣style␣annotator␣module␣(Layser)`

Current Header

`␣Input␣Features.␣␣Layser's␣machine␣learning␣features␣represent␣the␣visual␣cues␣available␣in`

Name	Description	Value (prev. header - this header)
diffScope	lines header has immediate scope over	$\frac{2-28}{658} * 100$ - = -4
diffCase	line's orthographic case	140 - 108 = 32
diffPrevBlank	blank lines before header	1 - 1 = 0
diffNextBlank	blank lines after header	1 - 0 = 1
diffPosition	position in text	228 - 242 = -14
diffEndPunct	end punctuation type	2 - 0 = 2
diffAvgSpace	average intraline space between words	100 - 108 = -8
diffEmbedPunct	intraline punctuation	914 - 157 = 757
diffLftMargin	left margin	40 - 10 = 30
diffRgtMargin	right margin	390 - 900 = -510

Table 3: Final `Header` features, calculated as the difference between initial `Layser` features.

3.3 Lexical cohesion module (`Coheser`)

Relationships between words of a document are known factors contributing to its structure. This factor has been used widely in discourse structure segmentation in the form of thesaural relations (Morris and Hirst, 1991; Kozima, 1993), cue phrases (Littman, 1996), word association (Pereira et al., 1993), as well as with different types of token repetition (Hearst, 1993; Kan et al., 1998).

Lines containing prose (i.e. Main Content Text and Embedded Headers) by `Layser` are fed into the `Coheser` module for lexical cohesion analysis. In documents without formatting, the entire document will be passed to `Coheser` as content text, and the system behaves like similar lexical cohesion based segmentation systems. Like the `Header` module, `Coheser` generates a set of features, and leaves the learning to the final module.

Input lines are first assigned part of speech tags from COMLEX (Grishman et al., 1994), and closed class words are discarded. We use the remaining content words per line as a bag of words, maintaining the same granularity as in `Layser`. We compute a set of normalized similarity values between the source line's words and its neighboring non-blank lines (the target). To model the variance in cohesion strength over distance, we compute the same set of features between the source line's words and varying sized neighboring targets (1, 2 and 3 adjacent lines, both before and after the source line).

The machine learning features we compute are based on a battery of shallow indicators that have proved effective in detecting cohesion. We categorize these features into four groups:

1. **Repetition (4 features):** These include term type and stem form repetition. The term type features model noun repetition but are subclassed for different types (e.g., pronouns, common nouns and proper nouns), as they have been shown (Kan et al., 1998) to possess different cohesion strengths.
2. **Thesaural relations (8 features):** Similarity scores is calculated via w_{sim} (Resnik, 1995) for the two transitive, tree-structured relationships: noun and verb is-a (hypernym/hyponym) and noun part-of (holonym/meronym). Normalized counts provide separate features for the other WordNet relations.
3. **Word association (1 feature):** WordNet only captures word similarity when the words participate in a structured relationship. To find other non-structural relationships, we use a word association feature which measures the Dice coefficient of correlation between grammatical subject noun heads in a large corpus (Schiffman and McKeown, 2000).
4. **Cue word (1 feature):** This feature finds cue words at the beginning of the source line. The feature has seven possible values, corresponding to the six categories of cue words listed in (Cohen, 1984): 1) parallel 2) detail, 3) inference, 4) summary, 5) reformulation, 6) contrast, plus a value for no cue word.

Table 4 gives an example source line with adjacent target window along with sample values.

3.4 Feature weighter (`Combiner`)

For each line, we string together all the vectors that have been computed. Null fields are inserted for lines that do not have `Header` and/or `Coheser` features, ensuring that all vectors have the same length. To build the document structure tree, we run the two tasks in series: we re-run the line annotation task with the combined features and then run the hierarchical segmentation task.

We used the `Ripper` (Cohen, 1995) machine learner to generate the two classifiers. `Ripper` outputs a human readable hypothesis file, which we used to validate whether certain common sense rules were learned by the system. Figure 3 shows some sample rules.

Sample logical style rules:
- **if the next line is indented and looks like a list, then it's a** *List Item*.
- **if the average intraline spacing is high and it's not in the first 40% of the document, then it's a** *Table*.
- **if the line is preceded by blank lines and the line length is short and within the first 60% of the document, then it's a** *Header*.

Sample segmentation rule for documents with headers:
- **if the difference in the embedded punctuation feature is greater than 77, then the nesting level is** *increased by one*.

Sample segmentation rule for documents without headers:
- **if the previous line is blank and the next line's word association score on a window of five lines is small and the previous line's noun hypernym word similarity on a window of three lines is high, then this line is a** *header at the same priority as the last one*.

Figure 3: 5 translated rules from `Combiner`.

4 Evaluation

We used seven documents, all from the medical domain (journal papers, health information book chapters and patient medical records), either converted from HTML or originally text, containing a total of 2515 non-blank lines. Eleven human subjects, all graduate students, given standardized instructions and no time limit, provided annotations for these documents. For logical line style, annotators categorized input text lines as one of the 11 logical styles. For hierarchical segmentation, subjects added nesting depth annotations to lines that they marked as Headers or Embedded Headers.

We also evaluated CLASP's performance on the task of linear segmentation, judged at paragraph boundaries. To do this while reusing the human judgments, we removed all headers from the corpus to create a new headerless corpus, and reassociated the nesting annotations with the lines after headers as segment breaks.

The gold standard was established for all lines that had a majority of annotators agree on the line style or nesting depth. Average human precision, Kappa agreement, and baselines (all Main Content lines for the logical style task, all sections at the same level for the hierarchical segmentation task, no segment break for the linear segmentation task) were calculated for all gold standard lines. We then ran CLASP using 5-fold cross validation to find the system's performance. Table 5 gives figures for this glass box evaluation of the system.

The results are very promising. With respect to the gold standard majority, CLASP performs above the level of the average human annotator, except for in the linear segmentation task. To further analyze CLASP's performance, we assess the features used by `Ripper`, since it implicitly does feature selection when constructing its hypothesis. In the line style task, many of the lexical cohesion features replaced layout ones as more accurate indicators of logical style, making up 17% of the `Combiner` learned conditions, while shrinking the ruleset by 8% over the initial `Layser` one while increasing precision. Document features were not used much, as the corpus did not contain any exceptional documents (ones without certain features, such as a caseless email), but all other layout features targeted specific logical styles such that they proved useful.

In hierarchical segmentation, CLASP relies heavily on the difference features provided by `Header`, accounting for 10 of 24 conditions. Remaining conditions used hypernym and holonym word similarity as well as lemma repetition. The linear segmentation task is more difficult, since headers were removed. The rule base for this task is smaller (4 rules, 11 conditions), and word association and line position substitute for the unavailable `Header` features. As the system's performance is worse than in the task with headers, we can assume `Header` difference information is more valuable than lexical cohesion, which in turn is more useful than word association. Cue words and other count-based repetition features were pruned, due to sparse data.

The architecture of CLASP splits the document map task into two different subtasks, that of line style and nesting. This distinction is real and is evidenced by the difference in Kappa. Also, annotation of hierarchical segmentation is more difficult and less certain on deeper levels: 91% agreement on level 1 headers, and 81% agreement on level 2, 3, 4 headers. For the hierarchical segmentation, Kappa indicates only weak agreement (.35), but when we conflate all segment breaks into a single category in linear segmentation, Kappa rises to a strong .90. This is a stronger level of agreement than other reported work, but here we have access to headers.

We also performed a task based evaluation of CLASP, using its resulting document structure trees to assist genre classification. The CLASP system was trained using the seven training documents in the intrinsic evaluation, and then applied to new documents from the Heart Information Network[3]. These documents are classified into one of the eight resource types: newsletters, questions, resources, articles, recipe listings, directories, educational and newsgroups. We took five documents from each category and ran CLASP to derive all (5 * 8) = 40 document structure trees. We converted the document structure tree into features suitable for machine learning by taking the logical line styles and hierarchical segmentation and converting them into simple percentages, as shown in Figure 4. The performance of these features were compared to both using a bag of words as a feature (BoW), and to Karlgren and Cutting's (1994) features (KC) to perform classification. Table 6 shows the results of the different learning methods using n-fold cross validation.

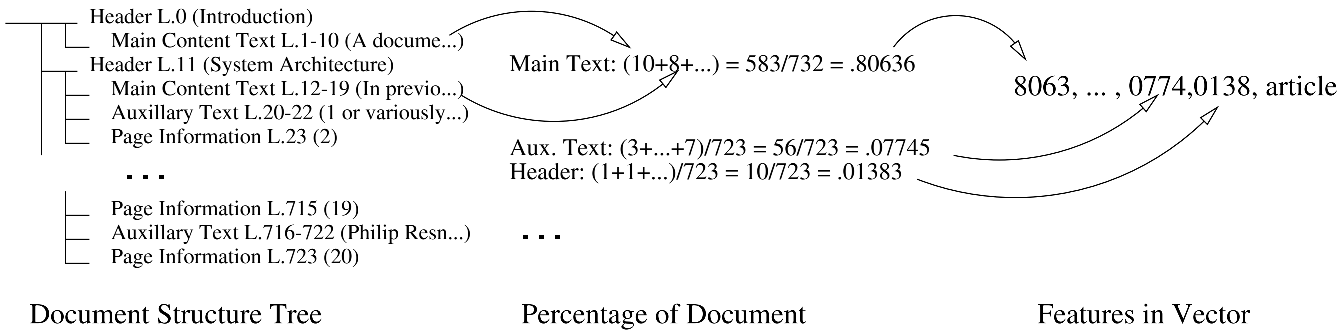

Figure 4: Converting a document structure tree to features.

[3] http://www.heartinfo.org/reviews/

The results are not conclusive, but the CLASP seems to outperform the other techniques. Human annotators agreed with the site's classification scheme much better on less ambiguous categories (e.g., recipe listings, newsgroups) than for more vague ones (e.g., educational, articles, resources). Surprisingly, using just the transformed document structure tree worked best. The conditions learned by CLASP's only model the vague classes which indicates that there is definite room to improve on the more concrete classes. Unfortunately, augmenting either the word based features or the Kalgren and Cutting inventory with CLASP features failed to improve performance.

4.1 Future Work

We currently do not deal with floating figures, tables and captions that are semantically related to a part of the text but which don't actually appear there (e.g., when a table is put at the end of an article as an appendix). To associate these items with their proper place, we can employ lexical cohesion to look for similarity between the figure and its referring text.

Style Catalog. CLASP represents a system to induce logical structure from physical markup, which makes it an analytical process. Reversing this process produces a logical to physical mapping that can be used in a text generation process. Given an ASCII document with a known document structure tree, we are working on inferring the feature values that are used within a document for a particular logical style (e.g., paragraph text is indented one tab and justified). The resulting *style catalog* can format new text, or reformat an existing document from one source to look like another.

5 Conclusion

We have integrated features from lexical cohesion and document recognition to produce document structure trees. CLASP extends the reach of text segmentation research by combining different paradigms in lexical cohesion, while adding additional features from layout when available. The base framework allows the system to deal seamlessly with documents with or without formatting information, and permits the eventual incorporation of additional features from richer sources of formatting markup than in plain text. In cases without formatting, we have shown that the system falls back to using lexical cohesion to perform segmentation.

In addition, in the task-based evaluation of CLASP, we demonstrated how document structure trees are directly applicable to current topics such as genre classification.

As richer formatted data becomes the norm in text processing, applications should make use of their input documents' structure, especially in sub-document processing; such as question-answering, document navigation and text summarization. CLASP allows applications to do this, and uses consistent framework and simple and robust design choices for granularity and extensibility.

References

Doug Beeferman, Adam Berger, and John Lafferty. 1997. Text segmentation using exponential models. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 35–46.

Hao Chen, Jianying Hu, and Richard W. Sproat. 1999. Integrating geometrical and linguistic analysis for e-mail signature block parsing. *ACM Transactions on Information Systems (TOIS)*, October.

Robin Cohen. 1984. A computational theory of the function of clue words in argument understanding. In *Proceedings of the 1984 International Computational Linguistics Conference (COLING 84)*, pages 251–255, California, USA.

William W. Cohen. 1995. Fast effective rule induction. In *Proc. 12th International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann.

Floriana Esposito, Donato Malerba, and Giovanni Semararo. 1994. Multistrategy learning for document recognition. *Applied Artificial Intelligence*, 8:33–84.

Ralph Grishman, Catherine Macleod, and Adam Meyers. 1994. Comlex syntax: Building a computational lexicon. In *Proceedings of COLING '94*, Kyoto, Japan.

Marti Hearst. 1993. Text tiling: A quantitative approach to discourse segmentation. Technical report, University of California, Berkeley, Sequoia.

Min-Yen Kan, Judith L. Klavans, and Kathleen R. McKeown. 1998. Linear segmentation and segment relevence. In *WVLC6*, pages 197–205, Montréal, Québec, Canada, August. ACL.

Bertin Klein and Peter Fankhauser. 1997. Error tolerant document structure analysis. *International Journal on Digital Libraries*, 1(4).

Hideki Kozima. 1993. Text segmentation based on similarity between words. In *Proceedings of the 31th Annual Meeting of the Association for Computational Linguistics*, pages 286–288, Columbus, OH, USA.

Diane J. Littman. 1996. Cue phrase classification using machine learning. *Journal of Artificial Intelligence Research*, 5:53–94.

A. Loeffen. 1994. Text databases; a survey of text models and systems. *SIGMOD Record*, 23(1):97–106.

Ch. Luc, M. Mojahid, J. Virbel, Cl. Garcia-Debanc, and M.-P. Pery-Woodley. 1999. A linguistic approach to some parameters of layout: A study of enumerations. In *Using Layout for the Generation, Understanding or Retrieval of Documents*, AAAI Fall Symposium, pages 20–29, North Falmouth, Massachusetts, November. AAAI.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–48.

Fernando Pereira, Natalie Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *Proceedings of the 31st Annual Meeting of the Association of Computational Linguistics*, pages 183–190, Columbus, Ohio, USA. ACL.

Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*.

Barry Schiffman and Kathleen R. McKeown. 2000. Experiments in automated lexicon building for text searching. To appear in the Proceedings of the International Conference on Computational Linguistics (Coling 2000).

Kristen Summers. 1995. Toward a taxonomy of logical document structures. Technical report, Cornell University, Ithaca, New York.

Mountaz Zizi and Michel Beaudouin-Lafon. 1995. Hypermedia exploration with interactive dynamic maps.

Source Line	`input features layser machine learning features represent visual cues available`
Target Window (2 lines before and after)	`translated features supervised learning layser module layser outputs application driven logical style explained later blank line` *source_line* `plain text documents categorize features five groups similar esposito shown table example values`

Name	Description	Sample Value
Repetition		
termPronoun	Pronoun repetition	0
termCommonN	Common noun repetition	$\frac{4}{10+29} * 1000 = 102$
termProperN	Proper noun repetition	$\frac{2}{10+29} * 1000 = 51$
lemmaRep	Simple token repetition (for all non-noun classes)	0
Thesaural		
wnSimIsA	Hyper/holonym similarity tree (using Resnik's W_{sim})	2821
wnSimPart	Holo/meronym similarity tree (again, via W_{sim})	155
wnAnt	Antonym (all classes) count	0
wnVerbCause	Verb "cause to" count	0
wnVerbEntail	Verb entailment count	0
wnNounAttr	Noun attribute count	0
wnPertain	Adjective pertaining to nouns count	0
Word Association		
wordAssoc	Word association (by Dice coefficient)	0
Cue Words		
cueWord	Type of cue word present at beginning of unit?	0 (no cue word)

Table 4: Lexical cohesion features and values calculated for sample source and target words used in the `Coheser` module.

Module	Task Type	Kappa	Baseline	$\overline{human}$ (σ_{xn})	5-fold C. V.
Layser	Line Style	.84	81.7%	90.1% (8.0%)	91.8% ± 0.6%
Combiner	Line Style				93.2% ± 0.6%
Combiner	Hierarchical Segmentation	.31	50.0%	53.4% (16.0%)	79.1% ± 6.6%
Combiner	Linear Seg. (at `P`s only)	.90	70.0%	92.2% (6.4%)	74.3% ± 1.6%

Table 5: Glass-box, intrinsic evaluation of CLASP. Figures reflect precision versus group majority.

Baseline	Kappa	$\overline{human}$ (σ_{xn})	(n = 40) fold C. V.		
			BoW	KC	CLASP
12.5%	.53	65.8% (2.7%)	20% ± 6.5%	17.5 ± 6.1%	22.5 ± 6.8%

Table 6: Black-box, extrinsic evaluation of CLASP via genre classification. Figures reflect precision versus group majority. BoW and KC figures are for both with and without the addition of CLASP features.