Layout and Language: An Efficient Algorithm for Detecting Text Blocks Based on Spatial and Linguistic Evidence

Matthew Hurst
IBM Research, Japan

ABSTRACT

The ability to accurately detect those areas in plain text documents that consist of contiguous text is an important pre-process to many applications. This paper introduces a novel method that uses both spatial and linguistic knowledge in an accurate manner to provide an initial analysis of the document. This initial analysis may then be extended to provide a complete analysis of the text areas in the document.

keywords: text block detection, tables.

1. INTRODUCTION

The Web has made us familiar with the problem of automatically processing huge volumes of marked up documents. However, there is still a vast amount of complex textual data recorded in a flat file format (with no explicit markup), or with only partial markup - SEC documents, plain text inserted into web pages, etc. Additionally, legacy and other documents with no electronic source require processing for inclusion in textual databases. In summary, there is a requirement for processing flat files, or parts of marked up files, whose complex layout structure is expressed implicitly via the physical location of its text blocks.

One of the preliminary tasks associated with these documents involves not the processing of the textual content, but the *location* of the textual content. Traditionally, this task is carried out using algorithms based on visual/spatial cues,[1] or on some abstraction of the text in terms of, for example, the location of tokens.[2] Both approaches may achieve reasonable results. However, as they exploit only a subset of the features (either white space or the 'shape' of the content) available they cannot deal with spatial ambiguities that may only be resolved by inspection of the linguistic content. The linguistic content (or the sub-linguistic description of the text in terms of characters) has been exploited in the classification problem where it is used to give cues as to the type of text block that has been recognized.[3] However, it it not generally considered a source of features for the problem of the *identification* of text blocks.

There are a number of text block detection and segmentation problems where an inspection of the language, i.e. the *content of the text*, is required before disambiguation may be achieved. Figure 14 presents a number of these ambiguity types. In addition to the problems involved in discovering coherent text blocks, purely spatial feature approaches will also encounter problems when using relative position (e.g. indentation) to derive the structure of the document when that indentation is not consistent.

The SEC table in Figure 1 details one type of ambiguity that may occur (multi-row cells (2) versus multiple cells). There are three places where there is the potential for a multi-row cell ('`Investments\ Medallion Loans`', '`Commercial Installment\ Loans`' and '`Unrealized depreciation of\ investments(11)`'). In the first case, the text blocks should not be merged as '`Investments`' is not a phrase that continues with '`Medallion Loans`' but is a superior cell that is also superior to '`Commercial Installment\ Loans`'. A superior cell is one that is encountered earlier than its inferiors in the access path when the table is being read.[4] In the second case and third case the text block should be merged.

In summary, there is a requirement for a text block detection methodology sensitive not only to the spatial characteristics of documents, but also the linguistic content. This paper proposes a framework and algorithms for such a methodology. The general method approaches the problem, in some sense, as a simulation of reading the text. Clearly, the amount of information required to perform such an analysis is greater than that for non-linguistic approaches due to the potential complexities of modeling language.

Correspondence: Email: `matt@trl.ibm.co.jp`; WWW: `http://www.cogsci.ed.ac.uk/~math/research/tables/`

```
BALANCE SHEET DATA
Investments
 Medallion Loans............  $45,642  $56,460  $66,437  $65,424  $66,338
 Commercial Installment
  Loans.....................   16,922   13,325   15,577   24,918   30,619
Unrealized depreciation of
 investments(11)............     (900)    (775)    (828)    (770)    (910)
                              -------  -------  -------  -------  -------
```

Figure 1. A table fragment with spatio-linguistic ambiguity.

```
                          Individual Grants                                Potential Realizable
                      Number of  Percent of                                Value at Assumed
                     Securities  Total Options                              Annual Rates of Stock
                     Underlying  Granted to                                 Price Appreciation
                       Options   Employees in   Exercise    Expiration          for Option Term
  Name                Granted(#) Fiscal Year    Price($/sh)    Date            5 %($)  10%($)
- ---------------    ----------- -----------   -----------   ----------    -------------------
  Steven H. Rothman    50,000      31.3%         $4.43         8/31/2001          $0    $30,000

  Howard Pavony        50,000      31.3%         $4.43         8/31/2001          $0    $30,000

  Robert Fries            -          -             -               -               $0   $0

  Ramon Mota            5,000       3.1%         $2.50         11/30/2001      $7,450   $12,650
```

Figure 2. An SEC table with Complex Header Material

The main benefit of this technique is to provide a high precision core analysis. In contexts where the potential for ambiguities arising from spatial features is high (such as tables and other formated text), this is extremely useful. In contexts where less informed approaches work with a reasonable amount of success already (such as paragraph text), it can provide a classifiable segment that may then be used to centre a heuristic or purely spatial approach with the prior knowledge of the type of text element being analyzed.

This paper outlines a principled and conservative approach to text block location. It is principled in that it uses the types of knowledge and contextual information used by humans. It is conservative in that makes decisions only in certain cases where high precision may be assumed. As a consequence, the method does not constitute a complete solution to the text block location problem. However, it does provide an initial analysis that may then be taken as input to any number of less restricted methods to complete the understanding of the document. The main contribution of this paper is the introduction of a methodology that integrates layout and language. A detailed presentation of an implementation of these ideas and an evaluation of the performance is given together with an overview of variations and extensions.

2. OVERVIEW OF ALGORITHM

The basic method employed by the text block detection algorithms presented in this paper consists of two stages. The first is to determine via spatial evidence the locations in the document where a text block may reach its boundary. When the text in a text block reaches a boundary, assuming it doesn't coincidently terminate, it wraps to the next available line within the text block. For English, this means that at the right boundary the flow of the text wraps and starts again on the next line at the left boundary. In Figure 2 this can be seen in the two cells below the spanning cell '`Individual Grants`'.

The ambiguities arise in the document when there is the *potential* for this wrapping to occur. At such locations in the text, the reader may either continue to the adjacent token on the same line if one exists, or wrap to a token on the line below if one exists. For example, in Figure 2, the text '`Number of`' may wrap to '`Securities`' or continue to '`Percent of`'. There may be more than one point in the text on the next line to which the text may wrap. Determining these points of ambiguity requires the use of spatial knowledge to indicate where they may occur. This spatial knowledge is essentially an encoding of our understanding of the devices used to lay text out on the page.

The second stage starts once we have a representation of these ambiguities. We must determine which blocks may be merged to form larger units of coherent text. There are a number of ways in which this can be carried out. The most precise is that which employs knowledge of the language in which the text is written. This application of the language model determines, where possible, if a link between two text blocks can be considered linguistically coherent. Merging block A with block B means that we accept that the text in block B is a valid extension of the text in block A.

3. A SIMPLE LANGUAGE MODEL

There are many formalisms and techniques for describing a language. The more linguistically formal methods lack the robustness that is required for a general purpose system such as that presented here. A more suitable system is the n-gram representation which captures via a simple statistic the validity of a sequence of words being part of the language. For example, the word sequence '`is required`' is likely to be judged as part of the language where as '`the the`' is not. The simplest form of n-gram suitable for making such judgments is the bi-gram. For the algorithm presented below, we wish to know if a pair of words can be considered as a valid sequence of words in the language. We represent this information by the tuple $\langle t_0, t_1, c\rangle$ where t_0 is the first token, t_1 the second and c the count. The count indicates the number of times that the bi-gram was observed within a corpus of a certain known size. This value is not exploited in the methods described in this paper but is included for completeness.

4. DOCUMENT REPRESENTATION

For NLP systems not concerned with the layout of the document, a suitable representation of the input text is a sequence of tokens (token instances). This sequence of tokens may be thought of as a degenerative graph, the arcs indicating the relation of sequence between tokens. However, as discussed above, when we consider the document as a spatio-linguistic object we must be able to represent the potential ambiguities where a number of possible continuations are presented for the flow of text. This can be achieved by using a graph whose nodes are text blocks which have already been determined and whose arcs indicate links between text blocks according to the flow of the text.

Another graph representing spaces is interlaced with this graph. Consequently any text block may observe its spatial context by inspecting the nodes in the space graph it is connected to to the left and to the right. The spaces in a document may form *streams* of space running between (potential) text blocks. A stream is a continuous vertical area of space with a minimum depth (the minimum stream size). These stream have length (the number of lines they continue for). A document may also include false streams where the position of the words in a paragraph, for example, cause an apparent stream of space to appear within the text block (see Figure 14 False Double Columns). In Figure 2, there is a stream between the two cells below the spanning cell '`Individual Grants`'.

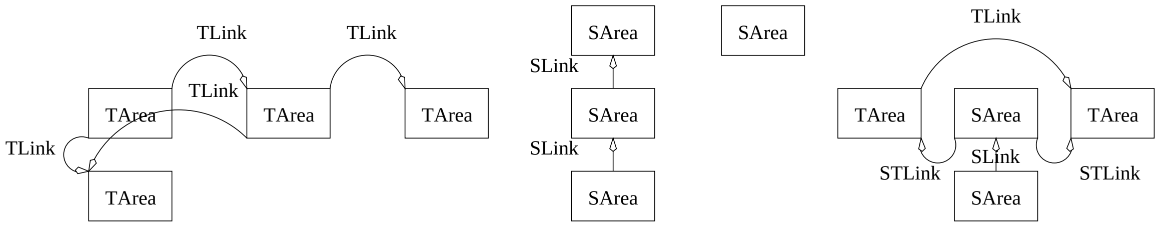


Figure 3. The text graph, the space graph, and a combined text and space graphs.

The following basic definitions indicate the interaction between the graph of spaces and the graph of text objects (see the illustration in Figure 3). The space graph has a set of nodes represented by **SArea**s and a set of arcs represented by **SLinks**. The text graph has a set of nodes represented by **TAreas** and a set of arcs represented by **TLinks**. The two graphs are related via a set of arcs from space to text areas represented by **STLinks**.

Area: is a tuple, $\langle x_0, y_0, x_1\rangle$ where x_0 is the left most x co-ordinate, x_1 the right most x co-ordinate and y_0 the y co-ordinate (an **Area** doesn't span multiple lines).

SArea: (a *space area*) is a tuple

$$\langle A, u, d, up, down, stream, left, right\rangle$$

where A is an **Area**, u is an integer representing the maximum length upstream and d is an integer representing the maximum length downstream, *up* is a set of **SLinks** sourced at this **SArea**, *down* is a set of **SLinks** sinked at this **SArea**, *stream* is a boolean value indicating whether this space is part of a valid stream or not, *left* is an **STLink** and *right* is an **STLinks** both sourced at this area and sinked at **TAreas** to the left and to the right respectively.

TArea: (a *text area*) is a tuple

$$\langle A, s_l, s_r, t_l, t_r, t_a, t_b, h, t\rangle$$

where A is an **area**, s_l and s_r are **STLinks**, t_l and t_r are **TLinks** (to the left and right), t_a and t_b are sets of **TLinks** (above and below). The last two fields are used to represent a **zone** - h is a pointer to a *head* (possibly null), t is a pointer to a *tail* (possibly null) - as described in Figure 4.

SLink: is a tuple $\langle source, sink\rangle$ where *source* is a pointer to an **SArea** which is the source of the link, and *sink* is a pointer to an **SArea** which is the sink of the link.

STLink: is a tuple $\langle source, sink\rangle$ where *source* is a pointer to an **SArea** which is the source of the link, and *sink* is a pointer to a **TArea** which is the sink of the link.

TLink: is a tuple $\langle source, sink, v, p, prev, next\rangle$ where *source* is a pointer to a **TArea** which is the source of the link, and *sink* is a pointer to a **TArea** which is the sink of the link, v is a representation of the validity of the link in terms of the language model and p is an integer representing the *perplexity* of the link. The *perplexity* of a **TLink** is computed by summing the number of **TLinks** exiting the source **TArea** of the link and the number of **TLinks** entering the sink **TArea** of the link. The **TLink** has an additional pair of fields which implement a linked list of **TLinks**. These fields are *prev* and *next* and point to the previous and next link with the same *perplexity*. These fields are used to maintain data structures for the efficient execution of the algorithm. The linked lists are handled by a list object which contains pointers to the head and tail of the list. These linked list objects are stored in a vector indexed by the perplexity of the links in the list. Two of these *link managers* are used, one for valid links and one for non-valid links.

zone: is a sequence of **TAreas** which are linked by a sequence of arcs (see Figure 4). The **zone** is not required to be rectilinear.

per zone link ambiguity: is a measure of the remaining ambiguities in the document and is the total of the number of **TLinks** exiting zones normalized by the number of zones in the document. Note that there may or may not be links exiting a zone in a completely and correctly analyzed documents depending on the parameters used to load in the document (the minimum stream size, the maximum vertical and horizontal distances between linked TAreas). Inspection of documents analyzed by hand suggests that per zone link ambiguity ranges from 0.8 to 1.7. This is the average number of links exiting a zone in the analyzed document. These links are non-valid links indicating that the spatial arrangement of the zones may have allowed a continuation but that that continuation has not been accepted by the analysis. Note that the per zone link ambiguity may be a good feature to use for the location of tables. Text in tables will have a higher per zone link ambiguity due to the increased number of streams.

The analysis of the graph is moderated by the perplexity of the links, as described below. A final data structure to accommodate this is an ordered set of lists of links. These lists are doubly linked lists implemented via the *prev* and *next* fields in the **TLink** objects. The lists are managed in such a way that any addition to the list is placed at the end. The algorithm may enumerate the items in the list from the start to the end. **TLinks** may be moved from one list to another, an operation carried out by removing the link from one list and adding it to the end of another.

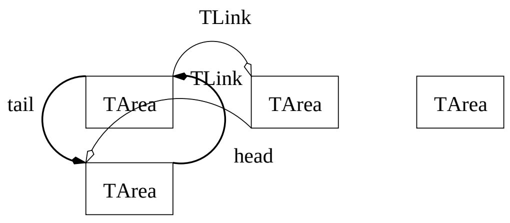


Figure 4. A **zone** is represented by a head **TArea** which a pointer to the tail which has a reciprocal pointer to the head. The internal **TArea**s must have exactly one exiting or entering arc (**TLink**) which describes the passage through the sequence of tokens of the text.

5. INITIALIZATION PROCEDURES

We consider the input to the algorithm to be a text file which may be described as a series of lines, each line is a series of characters of fixed width. The initialization of the data structures is carried out in the following manner.

1. The file is read in per line. The line is tokenized (breaking up the line into a sequence of spaces and character spans) and each token (**TArea**) and space (**SArea**) is added to the graph via the addition of **TLink**s and **SLink**s. The **SArea**s are linked to those **SAreas** in the line above which overlap horizontally, and to the **TArea** to the left if one exists. The **TArea**s are linked to the previous **TArea** on the line if one exists within a minimum distance and to the previous **SArea** on the line if one exists.
2. The **SArea**s are inspected to determine which are elements of a stream.
3. Once the file is read in, the **TArea**s are linked to the **TArea**s below and to the left indicating the potential flow of the text following a wrapping.
4. The perplexity of each links is computed. According to the perplexity, the link is added to the linked list of equally perplex links if there is one, or invokes the creation of a new linked list of which it is the head and tail. As the link has no validity at this point, it is recorded in the link set of non-valid links.
5. The initial set of zones is computed. This is carried out by locating the start and end of horizontal sequences of **TArea**s whose intermediate space areas not stream spaces, are of a minimal width or are non-existent. All **TLink**s entering **TArea**s other than the head of the zone, or exiting **TArea**s other the the tail of the zone are eliminated. The perplexity of the **TLink**s within the zone is set to 0. It is also practical to include a set of special 'splitting rules'. These are hand-coded rules that prevent the merging of certain sequences of character types. For example, the sequence `------ ------` is almost always used to underline elements in a table to indicate the width of their span.
6. The validity of each link with perplexity greater than 0 is computed. The validity is computed by determining if the last token in the source and the first token in the sink are a valid bi-gram according to the language model. If the link is determined to be valid, it is passed over to the link set for valid links.

The initialization phase results in two sets of lists of links, each set ordered by perplexity. The first set is the set of non-valid links. The second is the set of valid links. This initialization means that the main algorithm can now concentrate on only those links which are valid.

6. ALGORITHMS FOR BLOCK DETECTION

Having carried out the initialization process as described above, we now have a representation of the document in terms of the two graphs and two sets of links. The zoning process allows us to view the graph of **TArea**s as a graph of **zones**. We can characterize the input to the algorithm, then, as the tuple $\langle Z, V, I\rangle$ where Z is the set of zones (our view of the graph of **TAreas**), V is the set of valid **TLink**s ordered into lists of links with the same perplexity, and I is the set of non-valid **TLink**s.

```
link←list.head;
while(link){
     next←link.next;
     merge(link.source, link.sink);
     link←next;
}
```

ALGORITHM_0

```
for(p=3;p<=max_p();p++){
   for(p'=p;p>=2;p--){
      merge_count←merge_links_unique(p');
      if(p==p' AND merge_count==0){
        break;
      }
   }
}
```

ALGORITHM_1

Figure 5. Two text block merging algorithms.

The perplexity of a link indicates the ambiguity of its source and sink. The higher the perplexity, the more ambiguous. A perplexity of 2 (the minimum perplexity) indicates that there is one only link exiting the source and only one entering the sink, i.e. *this* link. Consequently, a valid link of perplexity 2 indicates that there is only one possible continuation of the text ending with the source, and that is to continue with the text at the sink *and* that this continuation is valid when the language model is inspected.

The `merge` operation is central to all methods described in this paper. Merging two zones can be achieved by a simple updating of their **TLinks** with new perplexity values, and reassigning their head and tail pointers as appropriate. However, is must be noted that merging two zones may result in the re-classification of a stream **SArea**. If the zones being merged break a stream such that the remaining stream segments are no longer greater than or equal to the minimum stream length, then the spaces in those streams of width 1 are remove and the neighbouring zones merged.

In summary, for each link simply merge the source and sink zones. Merging alters the characteristics of the link. As the link is now internal to a zone, its perplexity is set to 0 and it is removed from the set of perplexity 2 links by adjusting the linked list represented by its *prev* and *next* fields.

We define ALGORITHM_0 for this special sub-set of the set of V takes as input the `list` of valid **TLink**s with perplexity 2, as shown in Figure 5.

The second algorithm deals with valid links of perplexity 2 or greater. First, we should note that when a link's source and sink are merged, there are consequences for other links in the graph. As merging means that we are deciding which of the possible flows of text is the correct one, links exiting the source other than the link used in the merge are removed, as are those entering the sink. Consequently, the sinks and sources of those links respectively must adjust the perplexity of their other links. Figure 6 illustrates this process.

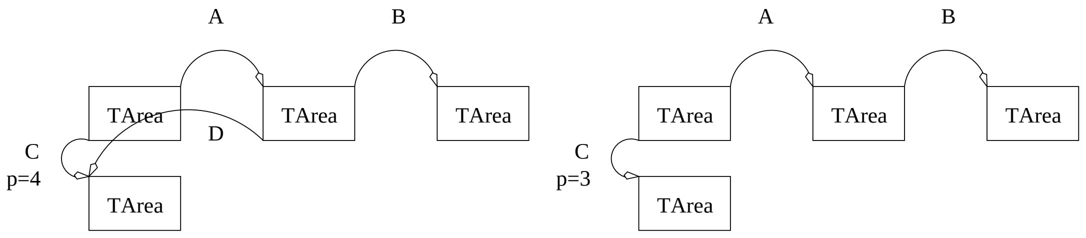


Figure 6. Merging the source and sink of link B means that link D is removed. This results in the reduction of the perplexity of link C from 4 to 3.

As a consequence of these alterations, the perplexity of some links is reduced. If the current list being dealt with is of perplexity p then some links will be reduced from perplexity $m > p$ to $m' > p$, some will be reduced from $m > p$ to $m' = p$, some from $m > p$ to $m' < p$, some from $m = p$ to $m' < p$ and some from $m < p$ to $m' < p$. Adjusting the perplexity means that the link moves from one list (with perplexity m) to another (with perplexity $m' < m$).

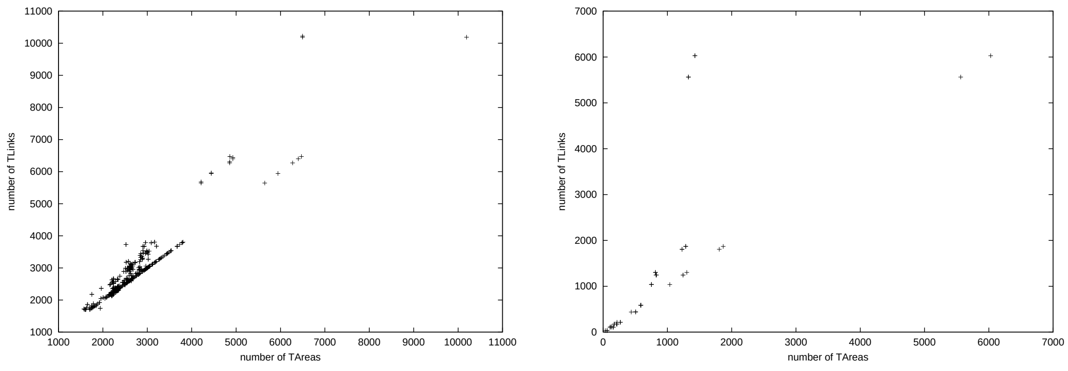
When a link is placed in a linked list it is added to the end of the list. There is a special case when the link pointed to by the *next* field of the current link moves to another list. In this case, following the *next* field would result in moving between lists with different perplexities. Consequently, in order to be able to enumerate a list of perplexity p while carrying out the merge operation, we must check for this case. If it occurs, we simply start again at the head of the list. This doesn't result in a loop as those links already processed have been removed from the list, and those which can't be processed are skipped as a matter of course.

A final detail is the notion of a unique valid link. When the perplexity is 2, the link is implicitly unique. However, when the perplexity is higher than 2 there may be more than 1 valid link exiting the source. This factor can be checked by inspecting the links exiting the source and entering the sink and computing the *valid perplexity* of the link. If it is 2, then the link is termed **unique**.

The second algorithm ALGORITHM_1 proceeds as shown in Figure 5. `merge_link_unique` proceeds in a manner similar to ALGORITHM_0 but with the check for valid perplexity and the check for displaced links.

7. PERFORMANCE ANALYSIS

Data Complexity The crucial element in the representation of the document is the set of links - especially the set of **TLink**s and valid **TLink**s. Investigating the number of **TLink**s in a document in terms of the document's size (which may be represented by the number of **TArea**s) shows that there is an approximate linear relationship. Figure 7 shows the number of TLinks with respect to the number of TAreas for documents with no tables, and that for a set of isolated tables. In both cases, the relationship is approximately linear. Documents containing tables will show a mixture of these ratios.



Documents with no tables. Tables extracted from documents.

Figure 7. The number of **TLink**s against the size of the document represented by the number of **TArea**s for documents containing no tables, and for tables extracted from their context.

Process Complexity Initializing the data structures when loading the document is $O(N)$ (the most complex part is computing the streams which can be implemented dynamically involving three passes of the document counting the number of spaces above, then below, then totalling). The basic case for merging is simply a matter of reassigning link information which may be done in linear time. However, if a link across a stream space is merged a check must be made to see if the stream has been fragmented into two streams which are no longer of the minimum length. If this is the case, the single space **SArea**s in the stream must be merged. As the number of such **SArea**s is at most $2(min\_stream - 1)$ this will result in a finite set of extra linear operations adding no additional factors to the complexity. Consequently, both of the conservative merging algorithms are dependent on the number of valid **TLinks**. It has been demonstrated that this directly proportional to the number of **TArea**s and consequently $O(N)$.

Evaluation of Language Model Size An evaluation may be carried out by considering the drop in per zone link ambiguity for the entire document. Inspecting the drop in link ambiguity shows the effect of the language model based algorithms on the analysis of the document. Figure 8 shows the average drop in link ambiguity as the size of the language model increases. The average initial ambiguity was 2.74 and the average ambiguity after processing as 2.16.

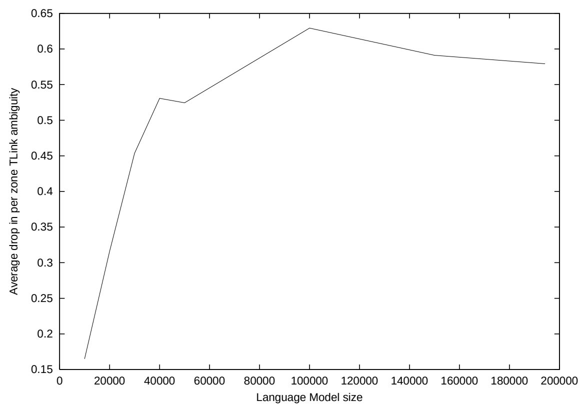


Figure 8. The average drop in per zone link ambiguity by the size of the language model

	no tables			tables			mixed		
	P	R	F	P	R	F	P	R	F
initial	99.15	61.20	75.68	99.61	74.34	85.14	99.40	65.46	78.94
`PARAGRAPH_MERGE`	87.70	90.10	88.83	75.42	99.11	85.66	72.18	99.53	83.38
`ALGORITHM_0`	99.05	61.77	76.09	99.61	74.62	85.33	99.37	65.82	79.19
`ALGORITHM_1`	98.57	75.93	85.78	98.67	85.47	91.60	98.68	81.64	89.35
`ALGORITHM_0`, `ALGORITHM_1`	98.52	75.91	85.75	98.67	85.47	91.60	98.68	81.64	89.35

Figure 9. Precision, Recall and F-measure results.

A larger language model will result in a larger number of valid links, however, it will also increase the number of links with a valid perplexity greater than 2, thus inhibiting the ability of the conservative algorithms described here to disambiguate.

Precision and Recall Evaluating the performance of the method described here and realized by the two algorithms in terms of the accuracy of the results produced by the implementation may be carried out with reference to a set of hand analyzed documents. The basic method of evaluation is to consider the ability of the system to decide between continuation points represented by multiple **TLinks** leaving a **TArea**. Links that have been selected as correct continuations, either due to the initialization algorithm, or due to the methods described, have a perplexity of 0. Other links have a perplexity of -1 (for rejected links) or > 1 (for links from or to ambiguous continuation points, or non-valid links). Consequently, we may use counts of the 0-perplexity links to compute precision and recall.

The table in Figure 9 shows the results for ALGORITHM_0 and ALGORITHM_1 run separately and serially, compared with the initial analysis of the document.[*] The results can be compared with the baseline provided by a naive algorithm (`PARAGRAPH_MERGE`) that merges text assuming that it is paragraph text. It merges block horizontally, then merges these line segments vertically.

As would be expected, the approaches incorporating linguistic information are more precise. In the sample of tables, where there are many instances of the problems described in Figure 14, the precision is far greater than the naive method.

Error Analysis There are two types of factors resulting in positive error. The first is a result of the assumptions and parameter settings of the initialization algorithm. The second results from incorrect merging of text blocks. (The combination of these two types, the merging of one or more incorrectly initialized text blocks, or text blocks resulting from errorful merges also exists.)

[*]This evaluation was carried out over a sample of 20 documents containing no tables, 20 tables extracted from their context and 10 document containing tables and non-tabular text, all taken from the 351 documents in the test corpus. The language model was extracted from the training corpus of 1246 documents and contained 194, 239 bi-grams (with 85, 566 unique tokens).

```
---------------------
December 31 January 31
---------------------
$ 43, 000   $ 120
$  9, 000   $ 215
```

Implied stream spaces captured by tabular artifacts.

```
1.1) xxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Embedded text block.

```
Embedded Section Heading xxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Embedded text block.

Figure 10. Captured Streams and Embedded Text Blocks.

In order to understand the first type, we must consider under what conditions incorrect initialization may occur. As the initialization algorithm only delivers **TAreas** and **SAreas** per line, segmentation in the vertical dimension can never be incorrect. Errors, then, may only occur when a space is not correctly identified - in fact, as the initialization algorithm segments all space bounded strings and then provides initial text blocks based on the nature of the spaces between these proto-blocks, errors occur when the spaces are incorrectly classified. There are three main causes of incorrectly classified spaces. The first occurs when a stream space is not recognized. This may happen if the stream is not of the minimal length, or if the space is captured in some textual device that separates it from the remainder of the stream. This may only happen in certain tabular environments such as the extract in Figure 10.

The other type of space based problem is really related to our notion about the semantics of the document. For example, in cases where there are embedded section headings, or enumerations (see Figure 10), if the analysis required their individual recognition, the segmentation of the text block would require a semantic recognition of the content, something that can only really be carried out in a post-process.

The third problem area is where there is no actual space between the end of one zone and the start of another. This can occur, for example in tables where the alignment, or mis-alignment of columns causes the streams to be 'squeezed' to zero width.

Incorrect merges occur either when there are a number of choices and the language model validates only one when in fact is should validate more than one, or when the language model incorrectly validates a link. Clearly these issues are to do with the quality of the language model, the appropriateness of the language model with respect to the document (sublanguages may have different and contradictory linguistic features) and the coverage of the language model.

8. EXTENSIONS

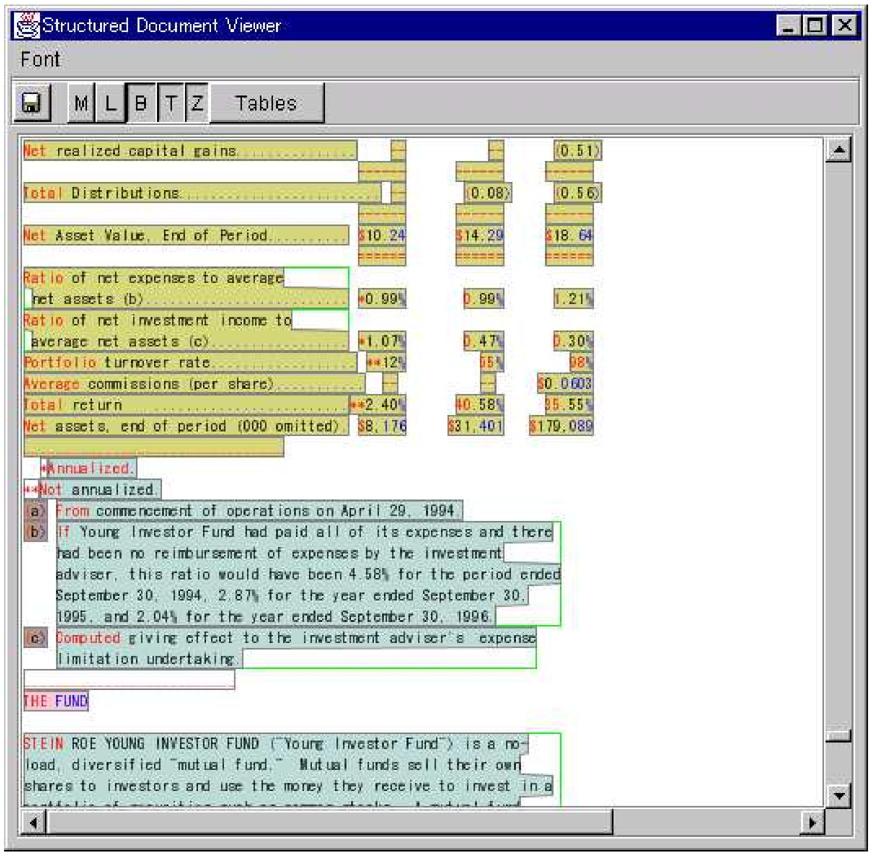
The algorithms and results presented thus far have been for conservative methods with high precision. The role of these methods in a document understanding system is to provide an accurate initial core analysis that may then be built on for a complete text block account of the document. There are two types of extensions that have been considered. The first deals with methods to take the original analysis and apply heuristics permitting further merge operations with a relaxed set of criteria. This type includes approaches which perform a classification of the current text blocks and then selectively performs merges according to the type assigned. One implemented method classifies paragraph fragments using a simple metric based on the location of the margins. Text blocks classified as paragraph are then expanded above and below checking for simple paragraph delimiters (empty lines or indentations).

The second type of extension is to consider variations on the type of language model used. As was noted earlier, the size of the language model may not positively effect the number of correct merges that the algorithm performs due to the increase in ambiguities. This problem may be reduced if a tri-gram model is used to judge such cases.

An even more precise language model may be employed if a well known and narrow domain is being investigated. For example, in the case of SEC documents, a dictionary, or glossary, of terms in the financial or economic domain may be used to perform merges prior to the application of the less precise bi-gram based approach. An experimental dictionary of 16018 terms, including 12187 phrasal entries, was compiled from online sources. A new merge algorithm (`D_MERGE`) was implemented that searched for paths in the document graph (the graph of `TLinks` over the set of `TAreas`) in a trie representation of the dictionary. This dictionary was used to create a new bi-gram model, resulting in 13255 bi-grams. A second dictionary and bi-gram model was constructed from a new sample of SEC documents.

	no tables			tables		
	P	R	F	P	R	F
`D_MERGE` (online sample)	99.17	62.61	76.76	99.38	75.64	85.90
`ALGORITHM_0, ALGORITHM_1` (online sample)	98.28	67.02	79.69	98.43	79.96	88.24
`D_MERGE` (sec table sample)	99.11	61.69	76.05	99.63	75.51	85.91
`ALGORITHM_0, ALGORITHM_1` (sec table sample)	98.58	63.38	77.15	99.40	77.95	87.38

Figure 11. Precision, Recall and F-measure results for experiments with language models created from a domain specific lexicon.



Net realized capital gains.............	—	—	(0.51)
	-----	-----	-----
Total Distributions......................	—	(0.08)	(0.56)
	-----	-----	-----
Net Asset Value, End of Period..........	$10.24	$14.29	$18.64
	=====	=====	=====
Ratio of net expenses to average net assets (b)......................	*0.99%	0.99%	1.21%
Ratio of net investment income to average net assets (c).............	*1.07%	0.47%	0.30%
Portfolio turnover rate..................	**12%	55%	98%
Average commissions (per share).........	—	—	$0.0603
Total return	**2.40%	40.58%	35.55%
Net assets, end of period (000 omitted)	$8,176	$31,401	$179,089

*Annualized.
**Not annualized.
(a) From commencement of operations on April 29, 1994
(b) If Young Investor Fund had paid all of its expenses and there had been no reimbursement of expenses by the investment adviser, this ratio would have been 4.58% for the period ended September 30, 1994, 2.87% for the year ended September 30, 1995, and 2.04% for the year ended September 30, 1996.
(c) Computed giving effect to the investment adviser's expense limitation undertaking.

THE FUND

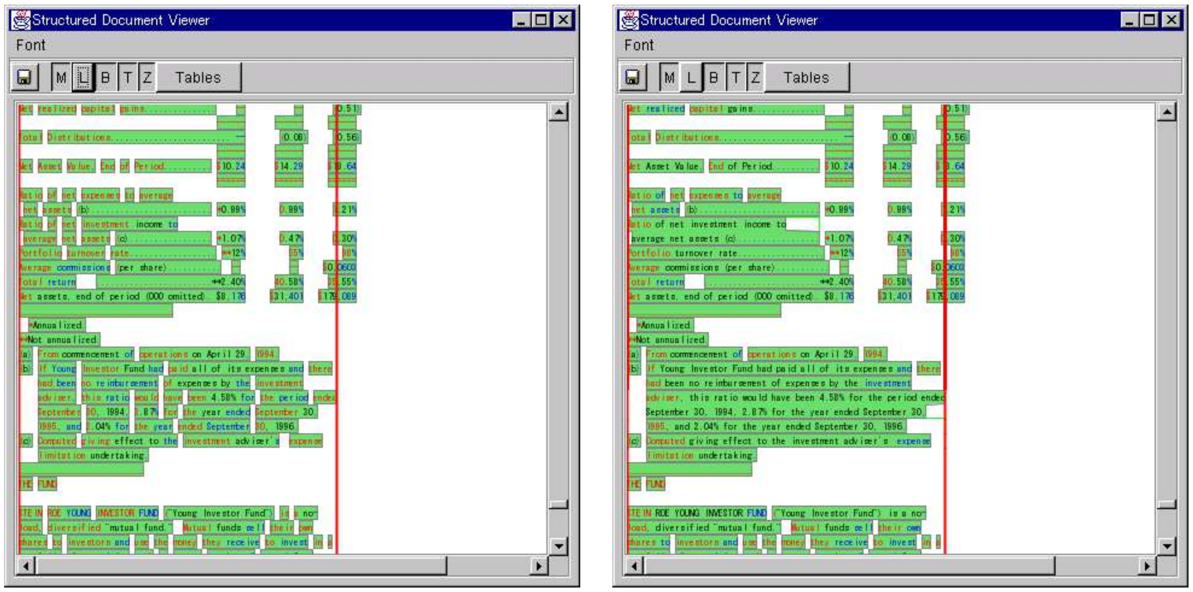
STEIN ROE YOUNG INVESTOR FUND ("Young Investor Fund") is a no-load, diversified "mutual fund." Mutual funds sell their own shares to investors and use the money they receive to invest in a

Figure 12. A portion of marked up SEC document containing part of a table (which exhibits some zero width clearance between columns and some mis-alignment), paragraph text with enumeration markers, a section heading and paragraph text.

The tables from these documents were extracted the phrases in the table were isolated. This smaller dictionary contained 566 entries, which resulted in 781 bi-grams.

The results of these experiments for document with and without tables are presented in Figure 11. As would be expected, the dictionary approaches are generally more accurate with lower recall. Also, the bi-gram model derived from the SEC tables is more accurate than the general bi-gram model.

What we can conclude from these experiments is that there is a precision based order to the types and sources of language model data. Dictionaries are more precise than bi-grams, and the bi-gram models are more precise when derived from sources closely tied to the domain. This ordering suggest a processing strategy that applies the more precise methods first and backs off through the different language models, through non-linguistic methods and finally applies simple heuristics. In addition the experiments have shown that tables are far more sensitive to the application of linguistic features.



```
et realized capital gains..............        —        —      (0.51)
otal Distributions.....................        —     (0.08)    (0.56)
et Asset Value, End of Period.........   $10.24    $14.29    $ 8.64
atio of net expenses to average
 net assets (b).......................   *0.99%     0.99%     1.21%
atio of net investment income to
 average net assets (c)...............   *1.07%     0.47%     0.30%
ortfolio turnover rate................   **12%       55%       68%
verage commissions (per share)........      —         —     $0.0600
otal return...........................  **2.40%    40.58%     5.55%
et assets, end of period (000 omitted)  $8,176    $31,401  $179,089

 *Annualized
**Not annualized
(a) From commencement of operations on April 29, 1994
(b) If Young Investor Fund had paid all of its expenses and there
    had been no reimbursement of expenses by the investment
    adviser, this ratio would have been 4.58% for the period ended
    September 30, 1994, 2.87% for the year ended September 30,
    1995, and 2.04% for the year ended September 30, 1996.
(c) Computed giving effect to the investment adviser's expense
    limitation undertaking.

HE FUND

TE IN ROE YOUNG INVESTOR FUND ("Young Investor Fund") is a no-
load, diversified "mutual fund." Mutual funds sell their own
shares to investors and use the money they receive to invest in a
```

Figure 13. A sample of an SEC document before and after processing.

9. CONCLUSION

This paper has presented a methodology for text block detection that is capable of representing and exploiting both spatial and linguistic information. The method uses conservative spatial knowledge to deliver a preliminary analysis and then extends ths analysis via the application of linguistic knowledge.

Acknowledgements

I would like to acknowledge the assistance of Yoshie Fujinuma who created the marked up data for the evaluation of this system.

REFERENCES

1. D. Rus and K. Summers, "Using white space for automated document structuring," Tech. Rep. TR94-1452, Cornell University, Department of Computer Science, July 1994.
2. T. Kieninger and A. Dengel, "A paper-to-html table converting system," in *Proceedings of Document Analysis Systems (DAS) 98*, (Nagano, Japan), November 1998.
3. K. M. Summers, *Automatic Discovery of Logical Document Structure*. PhD thesis, Cornell University, August 1998.
4. M. Hurst, *The Interpretation of Tables in Texts*. PhD thesis, University of Edinburgh, School of Cognitive Science, Informatics, University of Edinburgh, 2000.

Stream Ambiguities	Double Columns	For example, a paragraph Applying this of text is grammatically observation to the correct wherever the line segmentation of a double break occur. column of text will indicate where the line breaks occur.
	False Double Columns	Sometimes sentences may conspire to form false positives of rivers of white space which look like separated blocks but are in fact continuous text.
	Marginal Material	SETTLEMENT PROCEDURES For order of Book-Entry Notes solicited or an Agent and TIMETABLE: accepted or the issuer for settlement on the first
	Short Paragraphs	A short paragraph may be produce islands of text.
Line Spacing	Unmarked Headings	Adjustments upon changes in common stock In the event that the number of outstanding shares of Common Stock of the Company is changed by reason of recapitalization, reclassification, stock
	Double Spacing	We, the undersigned directors, attest to the correctness of this Report of Condition and declare that it has been examined by us, and to the best of our knowledge and
	Bulleted Lists	We may sell the securities: - - through underwriters, - - through agents or - - directly to a limited number of institutional purchasers or to a single purchaser.
Tables	Multi-column cells	Number Of Dogs Cats Horses Number Of Dogs | Cats | Horses
	Multi-row cells (1)	Date Of Name Birth Address Name | Date of Birth | Address
	Multi-row cells (2)	Weighted average number of common and common equivalent shares used in calculation $3, 926, 126
	Multiple cells	Costs and expenses: Cost of products sold 30,520,307 Technical personnel salaries 801,509 Selling, general and administratitive expenses 2,910,023 Interest expenses 5,898
	Continuities	Property, plant and equipment 1,504,809 Less accumulated deprecation 623,885
	Grid Quantization	Amount and Nature of Beneficial Ownership ----------------------- Name and Address Common of Beneficial Owner Shares ------------------- ----- Steven M. Rothman(2) 1,188,625 (3)(4)
	One-to-Many, Many-to-One	Unaudited September 30, March 31, Six Months Ended 1996 1996 September 30, (Dollars in thousands, 1995 1996 except current ratio data)

Figure 14. A catalog of spatio-linguistic problems.