Stylistic and Lexical Co-training for Web Block Classification

Chee How Lee
National University of Singapore
3 Science Drive 2
Singapore 117543
leecheeh@alumni.nus.edu.sg

Min-Yen Kan
National University of Singapore
3 Science Drive 2
Singapore 117543
kanmy@comp.nus.edu.sg

Sandra Lai
National University of Singapore
3 Science Drive 2
Singapore 117543
laihuiju@comp.nus.edu.sg

ABSTRACT
Many applications which use web data extract information from a limited number of regions on a web page. As such, web page division into blocks and the subsequent block classification have become a preprocessing step. We introduce PARCELS, an open-source, co-trained approach that performs classification based on separate stylistic and lexical views of the web page. Unlike previous work, PARCELS performs classification on fine-grained blocks. In addition to table-based layout, the system handles real-world pages which feature layout based on divisions and spans as well as stylistic inference for pages using cascaded style sheets. Our evaluation shows that the co-training process results in a reduction of 28.5% in error rate over a single-view classifier and that our approach is comparable to other state-of-the-art systems.

Categories and Subject Descriptors
I.7.m [**Document and Text Processing**]: Miscellaneous; H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia.

General Terms
Algorithms, Experimentation.

Keywords
PARCELS, co-training, lexical and stylistic learners, web page division, web page block classification.

1. INTRODUCTION
Extracting fields from web data is becoming an increasingly important issue, especially in cases where agents need to interact, *e.g.*, the Semantic Web. Unfortunately, web pages that use different HTML tags may result in similar layouts, and semantically similar pages may present information in different layouts. Due to this semi-structured nature of web pages and the mishmash of HTML coding, retrieving relevant information from web pages is a difficult task.

A possible step towards a solution is *web page subdivision and block classification*, in which a page is first divided into blocks, and the blocks classified by some scheme. Many algorithms benefit from using fine-grained blocks rather than uniformly processing the entire page. These include ad blocking, mobile device presentation and information extraction.

We present a PARser for Content Extraction and Layout Structure, or PARCELS, a system to perform the division and classification tasks using machine learning techniques. Unlike previous work, PARCELS uses a co-training model, adopting two independent views on block classification: one learner based on stylistic information and another based on lexical information. PARCELS is also designed to process real-world web page targets, handling more recent features of HTML including cascading style sheets (CSS) and non-tabular layout (e.g., `<SPAN>` and `<DIV>` tags).

In the following section, we discuss recent work in web page division/classification. Our method is based on co-training, which we review in Section 3, along with a description of the features used in the stylistic and lexical learners. In Section 7, we present our evaluation of PARCELS and compare it with previous work. We conclude with a discussion of the PARCELS software distribution, its associated utilities and availability.

2. RELATED WORK
The problem of decomposing web pages into blocks for post processing has been an area of recent interest in the literature. In our understanding of the published work, web page *fragments* [8], *blocks* [14], *elements* [16], *nodes* [9] and *shingles* all refer to the idea of tiling the physical representation of a web page with smaller blocks, as in Figure 3. In this paper, we follow the use in [1], using *blocks* to denote the divisions of a web page into semantic regions. These blocks often represent some logical division as governed by the application of interest, *e.g.*, an advertisement image for ad blocking, or the main contents of a page for adaptive content delivery.

To date, all related approaches that we have examined rely on the well-formed tree structure of the target page's hypertext markup. Although many real-world pages do not have well-formed markup, this is easily (and often) fixed by first canonicalizing the page using a tool such as HTML Tidy[1]. Typically, the canonicalized markup is used to form a Document Object Model (DOM) tree. The tree is then

[1] http://tidy.sourceforge.net/

manipulated directly to search for layout related structures (e.g., table cells and paragraph tags) or indirectly through rule-based post-processing [14][9]. PARCELS takes an identical pre-processing approach by working with the canonicalized DOM structure tree.

To our knowledge, all systems that have tackled the web page subdivision problem demonstrate their techniques on pages that do not utilize advanced HTML markup, avoiding pages that require Cascading Style Sheets (CSS) for correct font display and XHTML (e.g., `<DIV>` and `<SPAN>` tags) for advanced layout. We have conducted a diachronic survey of online news sites, and the results indicate that the use of such technologies is gaining popularity and thus important. A web page division system geared towards real-world use needs to cope with these possibilities.

Classification of blocks occurs next. In many ways, block classification is complementary to the problem of web information extraction (IE). In the latter, the goal is to identify blocks of a web page with a particular meaning; in the former, the goal is to classify which meaning a particular block has. IE techniques, such as wrapper induction, allow a database system to extract relational data from an external resource by automatically creating the appropriate contextual patterns. These patterns often contain structural HTML tags (paragraph and list item tags) as well as specific lexical items (*e.g.*, “price:”). Block classification algorithms also take both lexical and spatial information into consideration. We feel that structural and lexical information both play a large role in block classification, although they are largely independent of each other. Learning-based block classifiers that use both types of features should take advantage of this dichotomy, but to our knowledge, they currently do not.

Regardless of the division and classification approach, the comparative evaluation of approaches remains difficult. This is because division and block classification algorithms target different levels of granularity. A comparison requires a mapping of levels across systems. For example, the block model in [8] and [9] feature only a binary classification (relevant or irrelevant) whereas [13] makes a case for a three-class model of relevance. Since different applications may have different notions of relevance, we believe it is more useful to tackle the problem of block classification at a functional level (similar to Wong [14]) rather than assigning a numeric score of relevance.

Given these weaknesses in the current work, our paper makes two main contributions:

1. Our model explicitly takes advantage of the relative independence between structural and lexical information. We build upon prior work by introducing co-training as an additional layer in the classification process to achieve greater accuracy.
2. The division algorithm used in PARCELS explicitly handles both advanced markup types, incorporating useful CSS / XHTML features into the same generic framework for standard web page division, such as the ones based on table structures. Proper CSS parsing is non-trivial as attributes can be embedded within the tag or encoded within a style sheet (either internal or external to the page).

We also compare our work with a recently published system from the latest World Wide Web conference, with control for differences in division and block classification schemes. The results show our work comparable to the published system.

3. CO-TRAINING

Co-training [1], originally conceived for web page classification, is an iterative technique that decomposes a learning problem into two separate, independent views. Co-training is a bootstrapping process that gradually adds self-labeled data into the supervised training pool. Each view of the problem is represented by features that are used to induce a simple classifier. These classifiers are applied to unlabeled examples to annotate *k* examples from the unlabeled pool *u*. The self-labeled examples are added to the training pool for subsequent rounds and the two classifiers are re-trained for *i* iterations. Conditional independence of the views has been advised, but recent work by Nigam and Ghani [7] casts some doubt on these assumptions. In the original web page classification task, Blum and Mitchell [1] use features derived from the anchor text pointing to the target page for one view, and words on the target page as the other view.

To apply co-training to web block classification, we use two separate views based on the stylistic and lexical properties of blocks, as shown in Figure 1. This approach is motivated by the observation that lexical elements alone can often be effective in classification. So can structural elements: Shih and Karger [12] argue that the semantic structure of web pages is often obvious to users, even when written in an unfamiliar language. Note that the anchor text view used in Blum and Mitchell’s work [1] is not helpful in block classification, as links to specific web page blocks are uncommon.

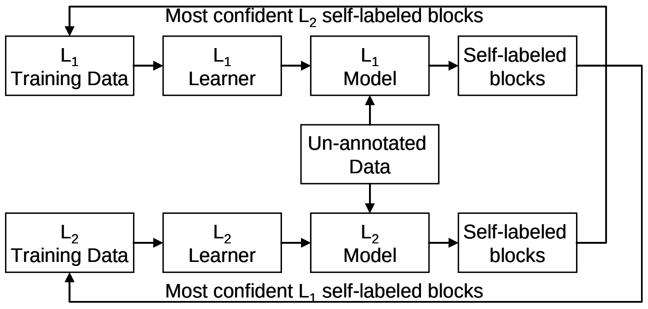

Figure 1. Co-training flowchart.

4. WEB PAGE DIVISION

We divide HTML tags into two categories for the purpose of web page division using the DOM tree. S*tructural tags* are used to define and group semantically-related text, and are intermediate nodes in the DOM tree. They subsume *content tags* which present the actual contents of the page. Tables, division and layer tags are thus structural tags whereas font formatting and list tags are content tags. In PARCELS – since our goal is fine-grained classification – paragraph, image and text that flank a content tag are also structural elements; i.e., they result in separate blocks.

PARCELS first does a pass over all of the leaf content tags, which are largely non-overlapping parts of the web page (with notable exception of layer tags and dynamic HTML). Content tags are then associated with a proper container (*i.e.*, the closest ancestral structural tag). Leaf content tags that are associated with the same container are merged, as long as they are adjacent in the DOM tree. This algorithm is shown in Figure 2. The calculation of a proper container makes our approach similar in spirit to Yin and Lee [16].

```
Input:   web page w;
         lists s and c of structural and content tags;
         maximum depth structural nesting depth d.
Output:  list of blocks B.

DOM_tree t := get_DOM_tree(tidy(w));
For each leaf node l (of type c) in t {
    Attach l to closest ancestor a (of type s) with max depth d;
}
For each intermediate node (of type s) {
    Block b := Merge all adjacent attached leaf nodes;
    B = B + b;
}
```

Figure 2: Web page division algorithm.

In HTML, control tags (*e.g.*, tables) can be recursively nested. In practice, we rarely see more than four levels of nesting. As such, we set a maximum nesting for structural tags, beyond which the tags are ignored. This allows deeply-nested text to be attached to higher-level ancestors, which we believe is more likely to provide coherent meaning. The result of such a division process is shown in Figure 3 below.

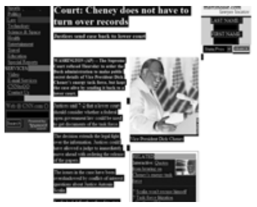

Figure 3: Web page after division. Contiguous dark regions indicate a single block.

Our division algorithm is simple and robust, and can be easily adapted to divide web pages according to other DOM-based methods. A different inventory of structural and content tags can be substituted to result in more coarse-grained blocks for comparison.

5. BLOCK CLASSIFICATION MODEL

Resulting blocks need to be labeled. The choice of appropriate labels is conditioned on the needs of the post-classification application and the expected input. We have chosen to classify online news pages by function. In a co-training framework, this corresponds to constructing two orthogonal sets of features for the separate learners l_1 and l_2. The choice of labels also influences feature design. We first discuss our classification scheme, and then turn to the features used in the stylistic and lexical learners.

5.1 News page block classification

There are three major reasons for our choice of online news as input:

1. Complex design. News sites are highly trafficked as they contain important and timely information to users. As such, news sites place an emphasis on the usability and density of information on their pages. These requirements are manifested in intricate layouts and result in tedious and human-incomprehensible HTML. News site layout also changes often, making other approaches such as wrapper induction particularly fragile. These qualities make the news domain an attractive target: 1) news itself is important to extract for information aggregation services, and 2) the induced rules may apply to news sites evolving over time and to other simpler web page domains.
2. Mixed levels of granularity. A highly-structured domain such as news allows us to experiment with different granularity in classification. News articles have specific features which are fine-grained fields that can be tagged, such as the article's location, publisher and reporter's name. The articles themselves are often formatted to allow easy access to key sections of related news. This phenomenon occurs in many well-structured text genres. We capture the phenomenon by annotating these mid-level tags: subheaders and supporting contents. Finally, coarse-grained tags apply to most web sites, as in the differentiation between site navigation and article-specific content. We have designed our classification with multiple levels of granularity to test PARCELS' ability to capture coarse-grained tags as well as fine-grained ones.
3. Related work. For the same reasons as the above, much previous work has been published on the extraction and tagging of news contents. Working with comparable input allows us to compare our methodology with previous approaches.

Label	Description
Main content	Main text of the article
Title	Headline of the article or phrase that summarizes the article
Site links / navigation	Links to other parts of the web site, including navigation bars, but unrelated to the article
Search	Text / links related to searching or search options
Links supporting contents of article	Hyperlink placed within the main content of the article
Supporting content	Content related to the main article, (e.g., captions, sidebars)
Supporting Image	Image related to the article's contents
Subheader	Embedded section headers internal to the main article
Site content	Content unrelated to the article (such as disclaimers, copyrights)
Site image	Site-specific image unrelated to the article
Advertisement*	External site advertisements. Not internal (e.g., link to related sections)
Links to related article*	Hyperlinks to other related articles (other reports on similar topics)
Newsletter*	Text / links related to newsletters or email alerts
Location*	Location where event occurred
Date / Time of article*	Date and time the article was published
Source station*	Source / provider / broadcast station
Reporter name*	Author of the article

Table 1: PARCELS news domain classification. Fine-grained classes are marked with the asterisk (*).

After examining over 10 online news sites, we defined 17 tags to model observed commonalities, as given in Table 1. As some blocks may demonstrate more than one class (*e.g.*, the lead paragraph of an article may contain the location where the event occurred), we asked annotators to mark the most salient one.

We labeled 20 documents from 15 different news web sites for classifier training, using a visual annotator (part of the PARCELS toolkit), giving a total of 1,625 labeled blocks. As one might expect, the distribution of the different tags in the data set was skewed considerably, with site navigation and main content blocks comprising almost 50% of the corpus. Details on the distribution of the corpus are given in Table 2. To create the unlabeled data set for our experiments, we further downloaded 50 news documents from 45 news web sites by selecting them from sources gathered from the Google News[2] service.

Class label	Frequency in corpus
Site Nav.	479 (29.5%)
Main Content	309 (19.0%)
Site Content	150 (9.2%)
Search	141 (8.6%)
Ads	134 (8.2%)
Related Links	118 (7.2%)
Site Images	74 (4.5%)
Supp. Content	63 (3.8%)
Subheaders	53 (3.2%)

Class label	Freq. in corpus
Newsletter	30 (1.8%)
Title	20 (1.2%)
Content Image	16 (0.9%)
Timestamp	14 (0.8%)
Reporter	9 (0.5%)
Country	6 (0.3%)
Source	6 (0.3%)
Content Link	3 (0.1%)

Table 2: Class distribution in the corpus, sorted by frequency.

5.2 Stylistic features

Stylistic features in PARCELS cover features that encode information about the block's placement on the page and its appearance. Our parser parses the stylistic properties of each block of text based on some of the design trends today [6]. We cover placement features first. Unlike some approaches that retrieve the *x*,*y* location of an element from a browser's internal DOM tree [16], our approach is browser-agnostic.

Authors of web pages have three major control structures to spatially lay out blocks on a stand-alone[3] web page: a default linear style, tabular formatting and layer formatting. As such, PARCELS first attempts to identify the prevalent system used. This is done by taking an inventory of tabular and division tags present in the topmost levels of the DOM tree. Our data analysis shows that `<TABLE>` and `<DIV>` tags used close to the root of the DOM tree have a strong correlation respectively to tabular and layer-based markup. This validates earlier findings [5].

[2] http://news.google.com

[3] Control structures that use multiple web pages, such as frames, are beyond the scope of the work presented here.

Once the layout control system is identified, an appropriate subsystem is called to parse the page. As both tabular and division-based layouts degenerate to the simple linear style when table, CSS and XHTML tags have been processed, both the tabular and division-based parsers internally invoke the linear parser for finer- grained blocks within the web page.

5.2.1 Linear structure

Paragraph (`<p>`), header (`<h1>-<h6>`) and rule tags (`<hr>`) specify the basic unit for a block of text. Consecutive paragraph tags constitute separate blocks for our task, in contrast to earlier work which tends to view a span of paragraphs as a single block. The resulting fine-grained blocks are necessary for semantic classification. Use of coarser-grained blocks tends to conflate many classes. The power of fine granularity classification comes with a price: the number of words in an average block is much lower than in other division models, which affects the classification power of the lexical view in the learning model.

5.2.2 Table Structure

Tables are the most widely used control structure on the Web. Initially introduced to present data in tabular format, the table has evolved into a formidable control structure and has been extensively analyzed by researchers.

Hurst and his colleagues have analyzed data tables in scientific texts [2][3][4]. We follow their formalism by introducing features that model the 1) reading order (or cell flow) of table elements, 2) the implied semantics of adjacent data values and 3) the implied semantics of table cell positions:

1. Features for cell flow refer to how each cell is positioned next to another in their parent table and how each table is positioned within the nested parent table. These features may provide extra information that help to classify cells correctly.
2. In [3], Hurst implies that the data of neighboring table cells can help determine a particular cell's data type. As cells of the same data type are often closely related, this affects how a table as a whole is intended to be read. In PARCELS, we model this by introducing a feature that groups cells in the same row or column together if they have similar data types. Data type in this context is determined by the cell's type of data and its word density, where applicable. Type of data is determined by the text in the cell. If the text in a particular cell is all numeric, it would be different from another group of cells which are words. Similarly, word density refers to the number of words that particular cell contains with respect to the whole page. It indicates the importance of a cell in terms of word density.
3. The position of table cells is equally important. We have created features that encode each cell's visual position. These features enable us to infer the cell's position with respect to any nested table or the parent table itself. Using the position of the cells and their stylistic properties, we can determine whether a row or a column of cells are header cells and such. Finally, as nested tables are a very common phenomenon in this process, we have added an additional feature to model cell depth.

5.2.3 XHTML/CSS Structure

In contrast to table cells, divisions and spans can be laid out independently. These structures are often used to impose logical structure and reading order on the elements of the page. CSS tags allow both division and spans to inherit either relative or absolute positioning features that also need to be encoded. For pages that use these tags, percentages as well as absolute pixel settings can be used for setting the height or the width of divisions. Absolutie pixel positions need to be converted to percentages. This is done by finding the maximal *x*,*y* position for any cell on the page, and normalizing the position of all other cells against this coordinate.

XHTML also allows the layering of overlapping elements using a CSS attribute, `z-index`. Overlap affects classification as content is placed over background images, and advertisements occasionally float over the main page. As such, the z-index is also modeled in our layout features.

5.2.4 Font features

The textual formatting of text may affect its classification. For each block, we derived features that model its words' color, weight, family, size and hyperlink features. Instead of capturing their value directly, we model the relative difference from the page's median values. For example, we target to learn that larger fonts than normal (rather than learn that a specific font size) indicate subheaders and titles. Learning relative differences helps to make the learned model more portable to new, unseen web pages.

5.2.5 Image features

Web blocks often feature images in the form of either web page decorations or content features. We model their sizes (where available) and number of images within a block.

5.3 Lexical Features

The lexical view receives only the string of tokens present in a web page block. PARCELS first calculates low-level features based on the words themselves and their statistical properties. To model the linguistic and functional properties of the web site, PARCELS also calculates high-level features, consisting of the part-of-speech (POS), types of hyperlinks and image counts for a span of text.

5.3.1 Low-level features

We have provided the learner with two basic features to model the count and vocabulary of the words present in the text block. We first extract text that appears in the browser by extracting any `alt` text from image tags and then removing any remaining HTML tags. The resulting words are stemmed using Porter's stemmer and the stems are used as features for classification. Fine-grained classes (*e.g.*, newsletter and search) often have a set expression or vocabulary that makes them distinguishable. The weight of each word is set in accordance to its TF×IDF (term frequency × inverse document frequency) [10] score, in which the IDF component has been pre-computed from a separate corpus.

5.3.2 High-level features

To perform part-of-speech (POS) tagging, we utilize QTAG[4], which uses a simplified variant of the common Penn Treebank tag set. Ratios of each part of speech tag count relative to the total are provided. This results in 32 POS-related features. For certain common POS, we provide aggregate POS features as well (e.g., a noun ratio adds the separate plural and singular noun ratios together).

The lexical features include four link-related features: *mailto-links*, *image-links*, *text-links* and *total-links*. *mailto-links* refers to the number of links containing the string "mailto:" which implies feedback / email functionality. *image-links* refers to links which point to images (quite distinct from the stylistic vector's image tag count). This can help identify thumbnails that point to larger versions. The remaining links are counted as normal *text-links*. The sum total of all three link types is given in the aggregate feature *total-links*.

6. CO-TRAINING WITH BOOSTEXTER

In performing co-training, an appropriate machine learner must be selected. In PARCELS, we employ Boostexter [11] as it is a multi-class learner that handles both numeric and textual features. Boostexter is an ensemble learning method which combines a set of weak classifiers to determine an input vector's classification. In a series of rounds, a new weak learner is induced over the training data, in which the tuples that have been incorrectly labeled from the previous round are more heavily weighted for the current round. Boostexter's weak learner corresponds to a decision stump, in which a single feature is used for discrimination.

As Blum and Mitchell's original work [1] is applicable only to binary classification problems, we have adapted the original algorithm to a multi-class setting. In the original algorithm, the number of self-labeled positive and negative examples that are added to the training data is determined by their initial distribution in the training data. This is to prevent skewing training data distribution during co-training iterations. In the multi-class problem, we also attempt to do the same, adding self-labeled instances from each class in the same proportion as in the initial distribution. We round up any fractional values to whole tuples, as many of the classes make up a small fraction of the corpus, and would otherwise not contribute a self-labeled example in subsequent rounds.

We make one further modification to the co-training algorithm. In the original algorithm, the *k* most confident self-labeled examples are replaced by *k* new examples from the unlabeled pool. As unlabeled data is inexpensive to prepare, we simplify the step by replacing the entire unlabeled pool of *u* examples at each round. Having a fresh set of unlabeled examples is likely to improve classifier accuracy (assuming the examples are drawn from the same distribution) as self-labeled examples that are not chosen at each round are not reconsidered in later iterations.

To create a combined classifier, the confidence of output labels of both the stylistic and lexical classifiers are compared, and the classifier with the higher confidence is used as the final label.

[4] http://web.bham.ac.uk/O.Mason/software/tagger

7. EVALUATION

To assess the performance of PARCELS and validate our claims, we have performed the following experiments:

1. We first assessed the co-trained classifiers with the scenario specified in the paper (e.g., using fine-grained block division, followed by block classification into the 17 categories. We compared the performance versus a single Boostexter-based classifier which used both the lexical and stylistic features together.
2. To assess how PARCELS performs on web pages that use advanced XHTML and CSS tags for layout, we conducted a separate evaluation with a corpus of web pages that specifically use CSS.
3. Finally, we compared our work with previous work which uses coarser-grained classification. In particular, we compared our work against another news web block classification system: Song *et al.*'s three-level model (*i.e.*, unrelated, topically related and important).

We used the standard measures of error rate to gauge the performance of the overall system, and the F_1 measure (defined as the harmonic mean between precision and recall) to capture performance on individual classes. We carried out all experiments using five-fold cross-validation, in which five independent training and testing splits were used to minimize noise. Throughout our evaluations, we determined an optimal number of rounds of boosting to be applied through cross-validation.

7.1 Basic Performance

For the first experiment, we assessed the classifiers' performance on the basic data set mentioned in Section 5.1. The default hypothesis corresponded to using the most frequent class in the training data as the label for every example. *Site navigation* was the most frequent class, corresponding to 29.4% of the training data. This resulted in a baseline error rate of 70.6%. Using the *single-view* model in which all features (both stylistic and lexical) were used, the learned Boostexter classifier improved the error rate to 35.0% percent.

Separate stylistic and lexical learners had access to only a subset of the combined classifier, and as such, we expected them to perform worse. Surprisingly, they both outperformed the single-view model, achieving 33.2% and 32.5% error for the lexical and stylistic classifiers alone (respectively). Also surprisingly, the combined classifier (which used the label of the most confident classifier) fared worse than either of its component classifiers and the single-view model, with a 38.4% error rate.

As mentioned earlier, co-training allowed us to incorporate self-labeled examples from unannotated data to improve performance on the test set. We used our previously downloaded set of 50 web pages for the co-training iterations, comprising slightly over 6,000 separate blocks. In each round, 300 blocks were self-labeled by the machine learners (this is the *u* parameter, the size of the unlabeled pool), and the 20 most confident examples (the *k* parameter) were added to the training data. These changes contributed at most a 20/1200 = 1.6% increase in the size of the training data per round. We co-trained for 20 rounds (parameter *r*) to exhaust the unlabeled data. The plot of the error rate versus co-training iterations is shown below in Figure 4. In the figure, the leftmost values correspond to the performance without co-training (round 0), and are the same as those mentioned in the above paragraph.

Similar to [1], we see that over a number of iterations, co-training did improve the performance of the classifier. All three learners benefited from the co-training process. The reduction in error rate for the classifiers was 35.8%, 6.1% and 34.8% for the lexical, stylistic and combined classifiers, respectively (averaged over five folds of cross validation). This result validates our hypothesis that co-training does improve learner performance in the domain of web block classification. The combined classifier achieved a 28.5% reduction in error rate in comparison with the single-view model. The lexical classifier exhibited both the smoothest and the best percentage improvement. We believe that this is due to the variety of different vocabulary exhibited on the web pages for each of the categories. The self-labeled data gathered from the stylistic learner may be able to help the lexical learner pinpoint key words that are discriminative.

How does PARCELS do on individual classes? We investigated the results on the combined classifier. In short, it performed satisfactorily on major classes, but disappointingly, it did not achieve our goal of fine-grained classification at any level. PARCELS is able to detect main content, site navigation and search bars well, but fails to annotate any of the fine-grained classes.

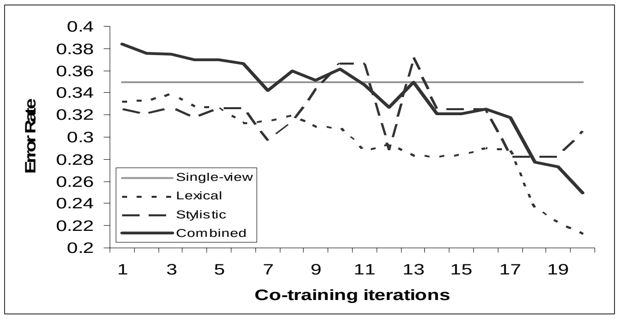

Figure 4: Error rate per co-training iteration for the learners with five-fold cross validation, for *i*=20, *k*=20, *u*=300.

Table 3 shows the effect of co-training on per-class detection. While co-training helped to improve the performance of the common classes, the performance of the smaller classes decreased.

Iteration 1	
Label	F_1
Main Content	.829
Site Navigation	.705
Search	.509
Site Content	.282
Related Links	.276
Site Image	.195
Content Image	.118

Iteration 20	
Label	F_1
Main Content	.892
Site Navigation	.816
Search	.703
Site Content	.148
Related Links	.071

Table 3: Per-class F_1 performance of the combined classifier, at iteration 1 (left column) and iteration 20 (right column). All other classes obtained an F_1 of 0.

We feel that the performance of PARCELS on coarse-grained tags is acceptable and serves as a motivation for us to further our work. PARCELS' failure to detect fine-grained classes forces us to re-evaluate our approach to fine-grained class detection. Perhaps part of the cause for the failure is the relatively small number of examples of fine-grained classes in the corpus. It is also likely that the feature set currently in PARCELS is better suited for coarse-grained classes. We hypothesize that contextual features (*i.e.*, features of neighboring blocks) and higher-level lexical features (*e.g.*, named entities for location and reporter name) would assist in the detection of these classes. Further modeling of features similar to those used in wrapper induction (*i.e.*, encoding of the XPath to the block) may also help address this limitation.

7.2 XHTML/CSS Performance

To assess PARCELS' performance on XHTML / CSS-based data, we performed a separate experiment using data gathered from news sites using XHTML and CSS-based layout. This corpus was smaller, comprising a total of five documents that were also manually annotated. This process resulted in 499 labeled blocks, with a distribution as shown in Table 4. We also downloaded a set of six unlabeled documents, resulting in 800 unannotated blocks to be used for unsupervised co-training.

Main content was the most frequent class, corresponding to 17.6% of the training data, and a baseline error of 83.4%. Using the single-view model in which all features (both stylistic and lexical) were used, the learned Boostexter classifier improved the error rate to 33.3%.

Class label	Freq. in corpus
Main Content	88 (17.6%)
Site Content	80 (16.0%)
Site Nav.	70 (14.0%)
Related Links	66 (13.2%)
Subheaders	39 (7.8%)
Content Links	33 (6.6%)
Site Images	30 (6.0%)
Ads	30 (6.0%)
Timestamp	21 (4.2%)

Class label	Freq. in corpus
Search	12 (2.4%)
Supp.Content	12 (2.4%)
Content Img.	6 (1.2%)
Title	4 (0.8%)
Newsletter	4 (0.8%)
Reporter	3 (0.6%)
Country	1 (0.2%)
Source	0 (0.0%)

Table 4: Class distribution in the XHTML/CSS corpus.

The separate stylistic and lexical engines performed worse than the single-view model before co-training, achieving error rates of 39.7% and 39.5% respectively for the stylistic and lexical views. The combined classifier again did poorer than either of its component classifiers, achieving an initial error rate of 42.3%. Using parameters similar to those in Section 7.1 to achieve the same 1.6% increase in training data size per iteration, co-training assisted in improving the overall accuracy of the separate learners but did not improve over the single-view model. One possible reason is that we simply did not have sufficient unlabeled data for co-training to work in this scenario. In the previous experiment, unlabeled data outnumbered labeled training data in a 4:1 ratio, whereas only a 2:1 ratio was achieved in this experiment. This is due to the difficulty in finding web sites that currently use the XHTML/CSS model for layout. We conjecture that a larger corpus of unlabeled data may reduce the error rate. Figure 5 shows the effect of co-training on error rate over nine iterations.

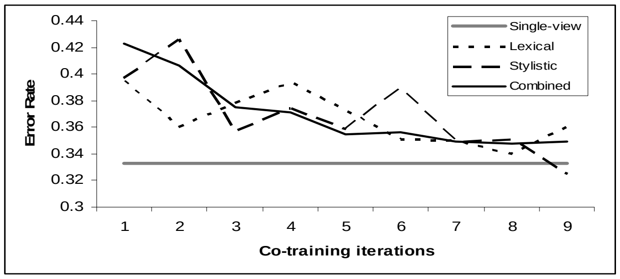

Figure 5: Error per co-training iteration for XHTML/CSS only data, where *i*=9, *k*=10 and *u*=70.

7.3 Comparative Performance

Song *et al.* [13] have also tackled a similar problem of web page block division and classification. Their final experiments classified blocks into a three-level model of block importance. The first level was used to annotate noisy information: advertisement, copyright and page decorations. The middle level (denoted as Level 2/3 in their paper) encompassed "useful information but not very relevant to the topic of the page", as well as other "relevant information to the theme of the page but not with prominent important". This included site navigation, related topics and topic indices. The final level labeled the most prominent parts of a page, such as headlines and main content.

For comparison, we modified our block segmentation algorithm to segment the page into a larger granularity similar to the original VIPS algorithm [18] used by Song *et al.* VIPS uses visually coherent parts in the page that sometimes are not possible to model using dominating nodes in the DOM tree. As a result, we could not divide the web page with the same model as theirs, but instead approximated their division algorithm by setting our system to create coarser-grained divisions.

Importance Level	Frequency in corpus
Level 1 (least important)	118 (65.9%)
Level 2	43 (24.0%)
Level 3 (most important)	18 (10.1%)

Table 5: Class distribution in the corpus using the three-level annotation model of Song *et al.* [13].

After segmentation, we re-annotated our training data according to their three-level system. The distribution of the three classes is shown in Table 5. With the annotated data set, we assessed the classifiers' performance. The default hypothesis, which predicted *Level 1* resulted in a baseline error of 34.1%. The single-view model using both stylistic and lexical features achieved an error rate of 19.5%, in comparison to Song *et al.*'s approach, which achieved an error rate of 14.1%, 18.9% and 16.2% using different machine learning models. As such, we believe our system is comparable, as we did not tune our system over through the application of different machine learners.

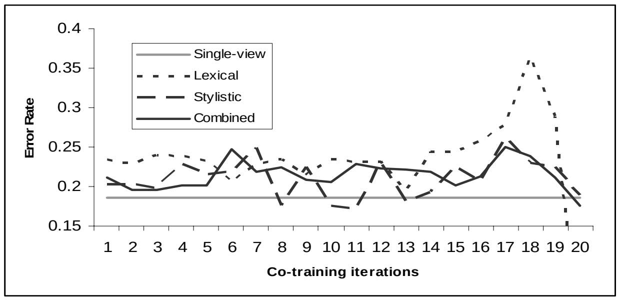

Figure 6:Error per co-training iteration on the three-level annotation model, using *i*=20, *k*=4 and *u*=40.

Co-training seems to be ineffective in this final scenario. We followed the same experimental procedure as in the previous experiments, keeping the ratio of self-labeled examples added to the training data pool balanced. However, the learners did benefit from the results of the co-training process. After 20 rounds of co-training that exhausted the unlabeled data set, we saw no apparent gain in performance; rather, the error rate increased. While the combined learner started with an error rate of 21.1% and ended with 17.6% error, the general trend over the iterations did indicate significant performance improvement. We plan on conducting further analysis to pinpoint the cause.

8. CONCLUSIONS AND FUTURE WORK

We have demonstrated PARCELS[5], an open-source, trainable system for web page division and block classification. We used a co-training model to leverage unlabeled data for partially unsupervised learning, and showed that it improves performance for our main task of block classification, achieving an overall error rate reduction of 28.5% over a single-view learner which used both stylistic and lexical features together in its feature set. Having separate co-trained learners improved performance on the basic stylistic and lexical learners for our fine-grained block division model. However, co-training failed to improve results over the single-view learner in our extended experiments. At this point, we have some plausible explanations why the co-training model did not improve upon the basic system, and plan to test these hypotheses in our on-going work.

Current work in PARCELS includes improving the quality of features used in classification. We plan to add modules to handle link structure [15] and common cross-document structure [8][17] to improve performance. Complex web sites often use content management systems that generate templatized HTML that is structurally common between web pages. Our planned module comprise of another view that models the web page in the context of its immediate community of pages.

Our current version of PARCELS is unable to perform fine-grained classification, in which specific fields, such as reporter's name, location and date/time are to be annotated. This tells us that our current framework is unsuited for information extraction use. We plan to focus on the annotation of these classes by using more powerful features such as those based on XPath, and using a machine learner that is more effective in modeling minority classes in multi-class scenarios (*e.g.*, cost factor settings in SVM).

[5] http://parcels.sourceforge.net/

9. REFERENCES

[1] Blum, A. and Mitchell, T. *Combining labeled and unlabeled data with co-training*. In Proc. of COLT '98, pages 92-100. 1998.

[2] Cohen, W.W., Hurst, M. and Jenson, L.S. (2002). *A flexible learning system for wrapping tables and lists in HTML documents.* In the 11th WWW Conference, Hawaii, 2002, pp. 232 – 241

[3] Hurst, M. and Douglas, S. (1997). *Layout & Language: Preliminary experiments in assigning logical structure to table cells.* In Proc. of Applied Natural Language Processing Conf., Washington, 1997, pages 217–220.

[4] Ghani, R. *Combining labeled and unlabeled data for text classification with a large number of categories.* In Proc. of IEEE Int'l Conf. on Data Mining, pages 597-598. 2001.

[5] Hurst, M. *Layout and Language: Beyond Simple Text for Information Interaction – Modeling the Table.* In the 2nd Int'l Conf. on Multi-model Interfaces, 1999.

[6] Ivory, M.Y. *Characteristics of Web Site Designs: Reality vs. Recommendation*. In Proc. of HCI Int'l Conf., 2003.

[7] Nigam, K. and Ghani, R. *Analyzing the effectiveness and applicability of Co-Training*. In Proc. of the 9th Int'l Conf. on Information and Knowledge Management, pages 86–93. 2000.

[8] Ramaswamy, L., Iyengar, A., Liu, L. and Douglis, F. *Automatic Detection of Fragments in Dynamically Generated Web Pages.* In Proc. of WWW '04, pages 443-454. New York, USA. May 2004.

[9] Reis, D., Golgher, P. B., da Silva, A. and Laender, A. H. F. *Automatic Web News Extraction Using Tree Edit Distance*. In Proc. of WWW '04, pages 502-511. New York, USA. May 2004.

[10] Salton, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*. 24:513-523. 1988.

[11] Schapire, R. E. and Singer, Y. BoosTexter: A boosting-based system for text categorization. *Machine Learning*, 39 (2/3), pages 135-168. 2000.

[12] Shih, L. K. and Karger, D. R. *Using URLs and Table Layout for Web Classification Tasks*. In Proc. of WWW '04, pages 193-202. New York, USA. May 2004.

[13] Song, R., Liu, H., Wen, J.-R. and Ma, W.-Y. *Learning Block Importance Models for Web Pages.* In Proc. of WWW '04, pages 203-211. New York, USA. May 2004.

[14] Wong, W. and Fu, A. W., *Finding Structure and Characteristics of Web Documents for Classification.* In Proc. of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD). Dallas, USA. 2000.

[15] Xi, W., Zhang, B., Chen, Z., Lu, Y., Yan, S. and Ma, W.-Y. *Link Fusion: A Unified Link Analysis Framework for Multi-Type Interrelated Data Objects*. In Proc. of WWW '04, pages 319-327. New York, USA. May 2004.

[16] Yin, X. and Lee, W.-S. *Using Link Analysis to Improve Layout on Mobile Devices*. In Proc. of WWW '04, pages 338-344. New York, USA. May 2004.

[17] Ye, S. and Chua, T.-S., *Detecting and Partitioning of Data Objects in Complex Web Pages*, In Proc. of Web Intelligence '04, Beijing. China. September 2004.

[18] Yu, S., Cai, D., Wen, J.-R., and Ma, W.-Y., *Improving pseudo-relevance feedback in web information retrieval using web page segmentation.* In Proc. of WWW '03, pages 11-18. Budapest, Hungary. May 2003.