Honours Year Project Report

PARCELS: PARser for Content Extraction and Logical Structure (Text Detection)

By

Lai Huijuan Sandra

Department of Computer Science

School of Computing

National University of Singapore

2003/2004

Honours Year Project Report

PARCELS: PARser for Content Extraction and Logical Structure (Text Detection)

By

Lai Huijuan Sandra

Department of Computer Science

School of Computing

National University of Singapore

2003/2004

Project No: H79110
Supervisor: Kan Min-Yen

Deliverables:
    Report: 1 Volume
    Program: 1 CD-ROM

Abstract

Because the flexibility of HTML allows it to be used differently to create the same effects, it is not possible sometimes to rely solely on the formatting or layout of a web page to find useful information. The Text Detection module of the PARCELS system was built to identify components of English news article web pages by performing lexical analysis on these components. A machine learning approach was adopted and BoosTexter, a boosting-based learner adapted for text categorization was used in this project. Lexical features for learning were chosen based on whether or not they could aid in distinguishing web page components in our chosen domain. Both generic features and specifically designed ones were used. An annotation system was also successfully built to collect data (annotated web pages) for the machine learner, despite the flexibility of HTML causing it to be complex and difficult to handle.

Subject Descriptors:

I.2.7 Natural Language Processing

Keywords:
Lexical analysis, content identification, text classification

Implementation Software and Hardware:
Java, HTML, Javascript, CSS, Perl, MS Windows

Table of Contents

Title i
Abstract iii

1 Introduction 1

2 Related Work 3

3 The PARCELS System 4
- 3.1 Choice of domain 4
- 3.2 PARCELS labels 5
- 3.3 Choice of machine learner 8
- 3.4 Lexical features 8

4 Annotation System 12
- 4.1 Overview 12
- 4.2 Features 13
- 4.3 Pre-processing of web pages 14
- 4.4 Post-processing of result files 15

5 Testing 16

6 Results 17

7 Conclusions 18
- 7.1 Discussion 18
- 7.2 Recommendations for future work 18

References 19

1. Introduction

The increasingly widespread use of the Internet has led to a growing number of people creating web pages, and because of the flexibility of HTML, it can be used differently to create the same effects on a web page. This only makes it more difficult for a machine to find useful pieces of information, though for the human user it may be easy to pick them out quickly by following the visual cues of the layout of the page.

However, if the layout of the page or style of the components on the page is unable to help the human or the machine, it will only be possible to find the required information by going through the text on the page.

For example, many web pages have a navigation bar running down the left-hand side, as a convention. In a particular web page for example, the navigation bar might be on the right-hand side instead, with advertising links taking up the left-hand side. In this case, we would not be able to rely on the layout of the page to find the navigation bar, but rather, examine the text of the links to identify what they are.

Here we describe the PARCELS (PARser for Content Extraction and Logical Structure) system, which is able to distinguish components of English news article web pages such as the above-mentioned navigation bar. In this project, we specifically describe the Text Detection module, which carries out this identification task without relying on formatting and using only lexical analysis with a machine learning approach.

The PARCELS system also contains a Stylistic Detection module (Lee, 2004) which uses a machine learner as well, and thus we are able to perform co-training with both learners. We will report the results of the Text Detection module alone, as well as compared with co-training.

This identification of web page components that we have carried out is useful for many information processing applications, including web page filtering, summarization or archiving, improving results for information retrieval and web searching, and reformatting for browsers in mobile devices.

The remainder of this report is organized as follows: some related work is highlighted in Chapter 2, followed by a description of the implementation of the PARCELS system, its labels, the lexical features used in the Text Detection module in Chapter 3. In Chapter 4, we describe the annotation system we have built to collect training data for our machine learner. Chapter 5 contains the details of our testing and the results are reported and analyzed in Chapter 6. Finally, in Chapter 7, we conclude with a discussion and some suggestions for future work.

2. Related Work

There has been a large amount of work done previously on the content extraction of web pages though most of these are based on the layout of web pages. Many varying representations of the layout of web pages have been researched, such as the widely used Document Object Model (DOM) tree, HTML Struct Tree (DiPasquo, 1998), and the VIPS algorithm for page segmentation (Yu, Cai, Wen, and Ma, 2003).

The content extraction system described by Gupta, Kaiser, Neistadt, and Grimm (2003) uses DOM tree representations of web pages, but it only makes use of the tree's ability to freely remove and modify specific nodes, and not the hierarchical structure of the tree (i.e. the web page's layout). Also, the system does not directly carry out analysis of the web page to find the content, but rather removes non-content using a set of filters, which mainly depend on the presence of certain tags or attributes within tags and less on lexical features of the text. This differs greatly from our system – we have utilized machine learning, which can be more easily generalized as opposed to hand-coded rules.

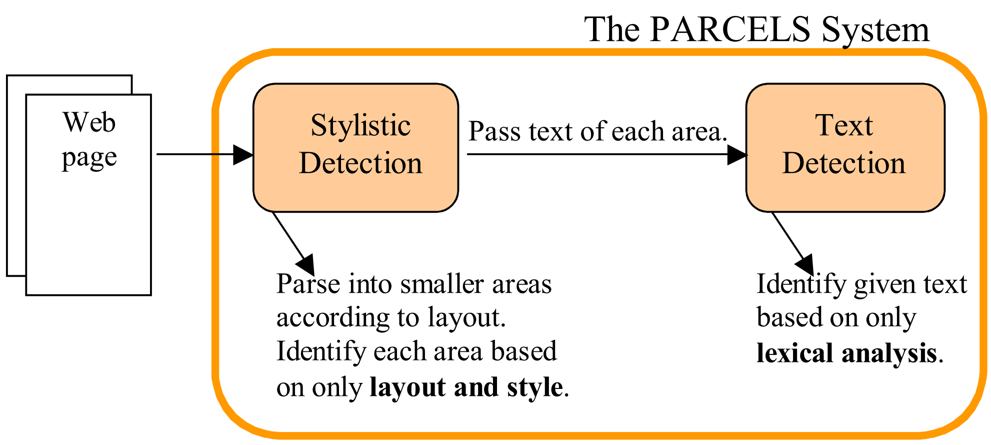
Less work has been done on content extraction using lexical features. One example is the article extraction module in the Columbia Newsblaster system, which extracts the article text from news article web pages (Evans and Klavans, 2003). As is the case with our system, it uses a machine learning approach with lexical features of the text. However, the only other web page components extracted during the process are the article title, images and image captions (with all other remaining components classified under a catch-all "Other" label). While this is sufficient for their system (a news article summarizer), we have gone one step further in including more labels, and labels which are more fine-grained.

Rather than content extraction, our problem may seem more similar to text categorization, i.e. assigning predefined category labels to documents, although clearly, assigning labels to entire documents is different from assigning labels to smaller blocks of text. A large number of learning approaches have already been applied to the text categorization problem (Yang and Liu, 1999).

Rhetorical Structure Theory (Mann and Thompson, 1988) is a method of describing the hierarchical structure of text via relations that hold between parts of the text. Though it is possible to carry out a RST structural analysis on news article web pages to find the semantic flow of information in the article, the lexical analysis and labeling of web page components we have performed is not sufficient to find such relations between these components. A deeper semantic analysis would be required to find relations whose presence may not be explicitly lexically signaled.

3. The PARCELS System

PARCELS stands for PARser for Content Extraction and Logical Structure, and it is so named because it is made up of two components: the Text Detection and Stylistic Detection modules, which parse web pages to perform content extraction and obtain logical structure respectively.


The PARCELS System
Web page
Stylistic Detection
Pass text of each area.
Text Detection
Parse into smaller areas according to layout. Identify each area based on only layout and style.
Identify given text based on only lexical analysis.

The way the PARCELS system works is as such: a web page is parsed by the Stylistic Detection module into many smaller areas according to the layout of the page. For each of these areas, the block of text contained inside is sent to the Text Detection module for identification, while the Stylistic Detection module also attempts to identify it without depending on the text but only its style and layout features. Thus the Text Detection module will be called repeatedly by the Stylistic Detection module in order for the former to identify all the components on a web page.

In this chapter, we explain our choice of domain for this project, the PARCELS labels used to identify web page components, the choice of machine learner for the Text Detection module, and lastly the lexical features that the Text Detection module uses to build a feature vector representations of text blocks.

3.1 Choice of domain

In order to keep the scope of this project manageable, we have chosen a specific domain of web pages to work with. Also, focusing on a single domain makes our task goal-driven in a way, as it requires us to consider beforehand a set of web page components specific to that domain which we want the system to target, instead of identifying general components that may not be of much practical use.

We selected the domain of English news article web pages for two main reasons. Firstly, there is a consistent set of components that are common to almost every news article web page, and these are discussed in detail in the following section. Secondly, with respect to the number of components on a page, news article web pages are lower in complexity compared to web pages from certain other domains. As one would intuitively expect, the bulk of such a page would be taken up by the main content text of the news article. However, on a product web page for example, there could be a larger number of components, such as the product name, its category, price, description, images, related products, reviews (which could consist of the reviewer's name, date, the given rating and the review text).

3.2 PARCELS labels

We propose the following set of 17 labels for the identification of components of news article web pages. In addition to the names and definitions of the labels, the table below also contains a few text examples of components each label represents. Examples such as images or components that are dependent on their location in the web page cannot be included in the table textually and have been described instead.

Label name	Definition	Examples
Title of article (parcels_title)	The headline of the article, a phrase that summarizes the article.	• Bush Turns to Key States to Break Deadlock • Next stop for Spirit: 'window' to Mars' interior
Date/Time of article (parcels_timestamp)	The date and/or time the article was published.	• 09:15 am ET • Last Updated: Tuesday, 6 January, 2004, 23:43 GMT • MAY 11, 2000 • Web posted at: 8:31 p.m. EDT (0031 GMT)
Reporter Name (parcels_reporter)	The name of the author or writer of the article.	• By Robert Roy Britt • Bijal P. Trivedi
Source Station (parcels_source)	The source/provider of the news article.	• Reuters • Associated Press • AFP • BusinessWeek Online • Chicago Tribune
Country where news occurred (parcels_country)	The country (or city, state) where the event described in the article took place.	• TIKRIT, Iraq • WASHINGTON

Image supporting contents of article (parcels_contentimage)	An image (a photograph, drawing, or sketch) related to the content of the article.	• A photo taken at the location where the event in the article had occurred • A photograph of a person mentioned in the article
Link supporting contents of article (parcels_contentlink)	A hyperlink located within the main content of the article.	Refer to definition.
Main content of article (parcels_maincontent)	The main text of the news article.	Refer to definition.
Supporting content of article (parcels_suppcontent)	Other text not belonging to the main text of the news article but are still related to the article or its content, including captions of images, captions of and information about audio/video files, any text in sidebars containing additional or related information about the article.	Refer to definition.
Subheaders (parcels_subheaders)	Text located within the main content of the article, which acts as a header for a portion of the main content.	Refer to definition.
Links to related articles (parcels_relatedlinks)	Links to other news articles which have similar or related content or articles previously reported on the same topic, and the date/time of these articles (if any).	• See also: Future Home Full of Web Wonders • Previous Stories: Mubarak receives written message from El-Bashir 2/26/2000 • Related News: AOL's Netscape sues Microsoft January 22, 2002
Newsletter (parcels_newsletter)	Text or links prompting the user to sign up for or subscribe to a newsletter.	• Subscribe to The Straits Times print edition today. In it you get exclusive reports, analyses and news packages. Do it by email or fax

		• CNET News.com Newsletters • Enter your email address here for our daily news update
Site images (parcels_siteimage)	Images that are part of the web site's design and are not related to the content of the news article.	• Site logo
Site content (parcels_sitecontent)	Text that is part of the web site's content and not related to the content of the news article.	• Note: Pages will open in a new browser window. External sites are not endorsed by CNN Interactive. • © 1999-2004 SPACE.com, Inc. ALL RIGHTS RESERVED. • Welcome to Wired News
Site links/navigation (parcels_sitenav)	Links to other parts of the web site not related to the content of the news article, also including links to other news sections (such as sports or world news) and navigation bars.	• Front Page • low graphics version | feedback | help • Business | Health | Science/Nature | Technology • about | terms of service | privacy | faq
Ads (parcels_ads)	Ads on the web site, including text ads and banner ads.	• Sponsored in part by Limited-time Holiday Offer INTERNET'S BEST PRICE • Make your company, and products a success. Special rate for new and small business. Inquire!
Search (parcels_search)	Text or links related to searching or search options.	• Search News • Advanced search options • search by keywords

The labels above can be divided into two main portions: five of them (Site images, Site content, Site links/navigation, Ads and Search) are general components which can appear on almost any type of web page, while the remaining 12 are specific to the news article domain. For the latter, we have attempted to come up with a comprehensive set of labels that can cover as many components as possible, and this was done by looking through web pages from a large number of news web sites.

The general labels serve to label the components on the web page that are not related to the news article. However, these labels are much less fine-grained, as our focus is still on the components specific to our chosen domain. For example, Site links/navigation covers any link not related to the contents of the article, such as navigation bars, as well as links to other parts of the web site, including other news sections.

We do not consider our identification task to be a multi-label one, therefore the set of components each label represents is mutually exclusive, and each component can only be tagged with at most one label.

3.3 Choice of machine learner

The performance of machine learned rules comes close to that of hand-coded rules (Soderland, Fisher, and Lehnert, 1997), and furthermore it is a less time-consuming approach which is more easily extensible and adaptable to other domains.

In this project, we have used BoosTexter (Schapire and Singer, 2000), a boosting-based learner which handles text categorization tasks, thus making it highly suitable for the Text Detection module. Boosting is a technique which combines many simple, moderately accurate classification rules ("weak" classifiers) into a single highly accurate rule. It has been applied to a number of tasks such as term extraction (Vivaldi, Màrquez, and Rodríguez, 2001) and text categorization using concept-based semantic features (Cai and Hofmann, 2003), giving significantly improved performance. As for BoosTexter itself, Schapire et al have evaluated its performance in comparison with a large number of other methods and results (Yang, 1997) and found BoosTexter's performance to be the best.

In addition, BoosTexter has many features that make its usage convenient as compared with other learners, and one of them is being able to handle multi-class problems. For a learner only able to handle binary class problems, we would have had to convert our task to multiple binary classification problems. BoosTexter also allows not only continuous-valued attributes, but also discrete-valued attributes (which take a value from a predetermined set), and even text strings.

3.4 Lexical features

The lexical features that make up the feature vector in our system were chosen mainly for the labels that they are able to distinguish. There are a total of 47 features, which can be divided into five main categories depending on how the values were calculated or obtained: TF*IDF-related, NE-related, POS-related, link-related, and contains-related. While some of the features are generic, some of them have been designed specifically to identify the web page components for our chosen domain. In this section, feature names are italicized and names of PARCELS labels are underlined for easy reference.

TF*IDF-related features

To obtain TF*IDF values for each word in the text block, stopwords (frequently occurring words) are first removed according to a list of 669 common words. The remaining words are stemmed using the Porter stemmer (Porter, 1980) and their term frequencies (TF) calculated. The TF value for each word is then multiplied with its corresponding inverse document frequency (IDF) value to give its TF*IDF value.

There are two TF*IDF-related features: *top-keywords* (text-valued attribute), which are the three words in the text block with the highest TF*IDF scores, and *total-words* (continuous-valued attribute), which is a count of all the words in the text block, including stopwords. *total-words* can help in distinguishing longer text blocks such as Main content of article and Supporting content of article from shorter ones like Title of article and Subheaders.

NE-related features

For Named Entity (NE) recognition, we have used the ANNIE information extraction system, which is part of GATE. There are six NE-related features: *total-ne*, corresponding to the total number of NEs present in the text block, and also *Person*, *Location*, *Organization*, *Date*, and *Time* (all continuous-valued attributes) which respectively refer to the number of these named entities present. We can easily see that these features can assist in identification of Date/Time of article, Reporter Name, Source Station, and Country where news occurred.

POS-related features

The text is first tokenized and then tagged using QTAG, a probabilistic parts-of-speech (POS) tagger, which uses a variant of the common Brown and Penn Treebank-style tagsets. There are 32 POS-related features, which include *total-tags*, the total number of tags the words in the text are tagged with, as well as *nouns*, *proper-nouns*, *pronouns*, *pronouns-indefinite*, *pronouns-wh*, *adjectives*, *adverbs*, *adverbs-wh*, *verbs*, *be's*, *do's*, *have's*, *conjunctions*, *determiners*, *determiners-pre*, *determiners-wh*, *prepositions*, *numbers*, *symbols*, *possessives*, *negatives*, *existential-there's*, *foreign-words*, *modal-auxiliaries*, *adverbial-particles*, *infinitive-markers*, and *interjections* (all continuous-valued attributes), which respectively refer to the ratios of the count of these tags to the total number of tags. Similar POS tags are grouped together so *nouns*, for example, is actually the ratio of the total count of both singular and plural nouns to the total number of tags.

The ratio of POS tags is useful for identifying components such as Title of article (it would most likely have a low ratio of pronouns like “he”, “she” and determiners like “the”, “a”) and also Reporter Name, Source Station, and Country where news occurred (all proper nouns).

Apart from the usual generic features associated with natural language processing (the above-mentioned TF*IDF calculation, NE recognition, POS tag ratio), we have also designed some other simple but practical features that can help in identifying news article web page components.

The POS-related ones that we created correspond to whether a particular sequence of POS tags is present: *sourcestation-string*, *country-string*, *country-sourcestation-string*, *sentence-begins-with-verb* (all discrete-valued attributes which take values false or true). *sourcestation-string* refers to whether there is a hyphen followed by a proper noun (singular or plural) at the end of a sentence. We created this feature to help distinguish Source Station components (e.g. “- AFP.” at the end of an article). *country-string* indicates the presence of a singular or plural proper noun followed by a hyphen at the beginning of a sentence, which can help identify Country where news occurred (e.g. “Singapore -” at the beginning of an article). *country-sourcestation-string* indicates the presence of “NP (NP) -” at the beginning of a sentence where NP is a singular or plural proper noun (e.g. Singapore (AFP) -” at the beginning of an article), which also helps to identify the two labels mentioned above. *sentence-begins-with-verb* refers to whether there is a verb in its base form at the beginning of a sentence, which can help in identifying Newsletter (e.g. “Subscribe to our...”).

Link-related features

We have created four link-related features: *anchors*, *email-links*, *image-links*, and *text-links* (all continuous-valued attributes). *anchors* is a count of the anchor tags (“<a>”s) and thus the number of links present in the text block. *email-links* refers to the number of links containing the string “mailto:” which implies that it contains an email address. This can assist in identifying Reporter Name components, as some of them are linked to the email address of the reporter. *image-links* refers to links which are images, and this can help in identifying Image supporting contents of article (most such images are thumbnails which when clicked on, lead to a larger image). The remaining links are classified as *text-links*, which are the most common, and can help identify Link supporting contents of article and Links to related articles. Site links/navigation and Ads can be either linked images or linked text, so both *image-links* and *text-links* will help in their identification.

Contains-related features

Contains-related features are named for the simple fact that they all indicate that the text contains some specific words. There are three such features: *contains-by*, *contains-newsletter*, and *contains-related* (all discrete-valued attributes which take values false or true).

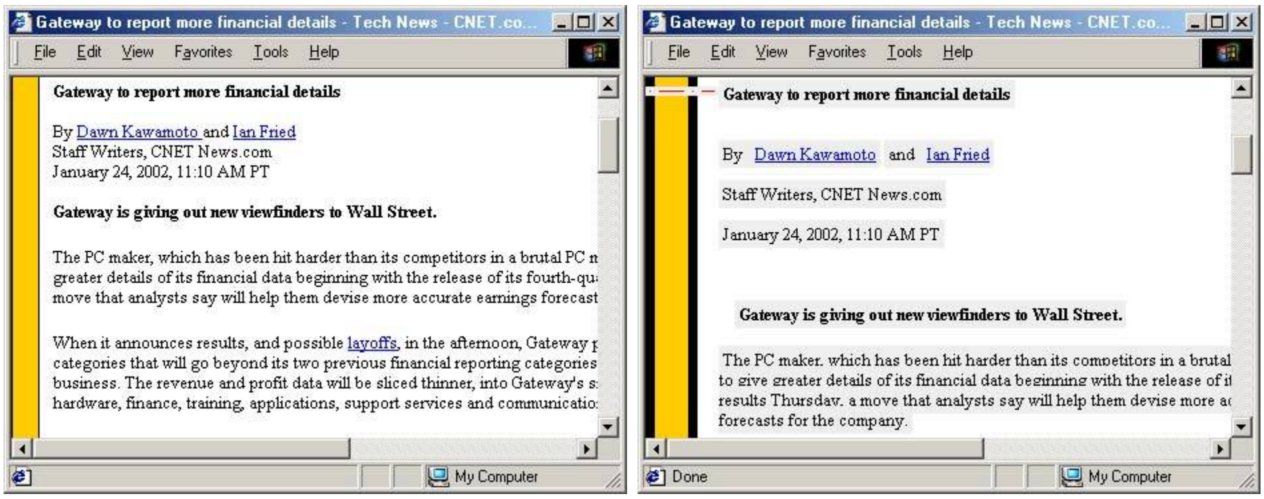
contains-by indicates the presence of "by" or "reported by", which can help in identifying Reporter Name. *contains-newsletter* refers to whether the text contains "sign up", "subscribe", "subscription", or "newsletter" or its synonyms as found in WordNet, and this helps in identifying Newsletter. Lastly, *contains-related* indicates the presence of "related", "previous" or "similar", as well as "articles" or "stories" or their WordNet synonyms, and this helps distinguish Links to related articles.

4. Annotation System

Using a machine learning approach requires us to obtain large amounts of training data in the form of news article web pages annotated with the PARCELS labels. As compared to manual marking up of hundreds of text blocks on paper, we decided to build an online annotation system. Being automated, it would save time and energy, but the trade-off would be the difficulties in building such a system as well as the possible loss of accuracy in annotation.

4.1 Overview

We first carry out pre-processing of each web page. The entire page is converted to an HTML form that can be submitted, and hidden input fields corresponding to each component are added to the form. Each component and its input field are given a component ID.


Gateway to report more financial details - Tech News - CNET.co...
File Edit View Favorites Tools Help
My Computer
Gateway to report more financial details - Tech News - CNET.co...
File Edit View Favorites Tools Help
Done
My Computer

Gateway to report more financial details

By Dawn Kawamoto and Ian Fried
Staff Writers, CNET News.com
January 24, 2002, 11:10 AM PT

Gateway is giving out new viewfinders to Wall Street.

The PC maker, which has been hit harder than its competitors in a brutal PC m
greater details of its financial data beginning with the release of its fourth-qu
move that analysts say will help them devise more accurate earnings forecast

When it announces results, and possible layoffs, in the afternoon, Gateway p
categories that will go beyond its two previous financial reporting categories
business. The revenue and profit data will be sliced thinner, into Gateway's s
hardware, finance, training, applications, support services and communicatio

Gateway to report more financial details

By Dawn Kawamoto and Ian Fried

Staff Writers, CNET News.com

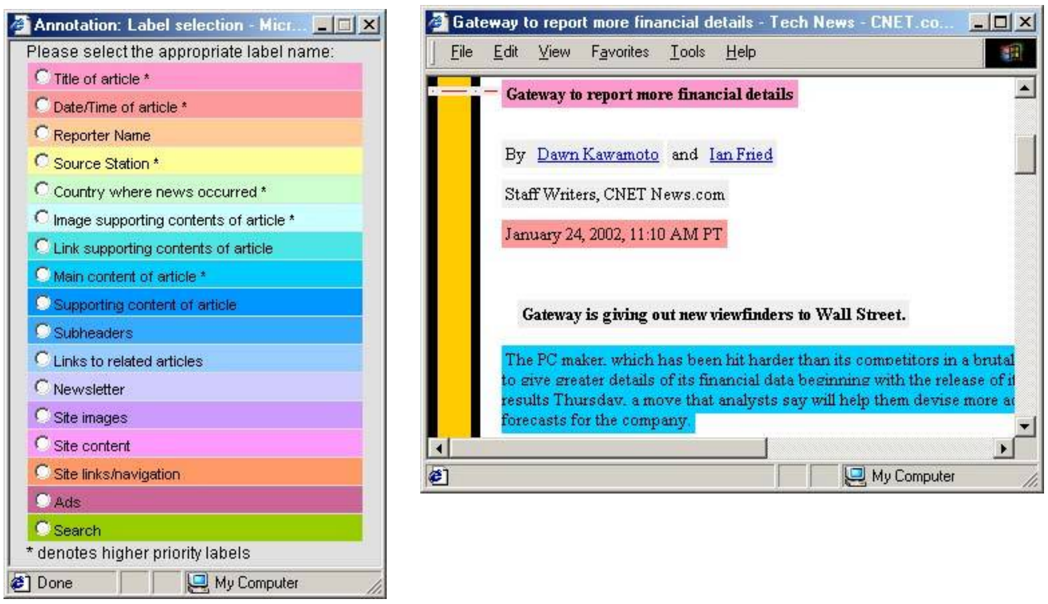
January 24, 2002, 11:10 AM PT

Gateway is giving out new viewfinders to Wall Street.

The PC maker, which has been hit harder than its competitors in a brutal
to give greater details of its financial data beginning with the release of it
results Thursday, a move that analysts say will help them devise more ac
forecasts for the company.

Figure 1. Before and after pre-processing.
Web page components are converted to light grey areas that can be clicked on and annotated.

When the human user clicks on a component, a popup box containing the PARCELS labels will appear. He or she can then select the most appropriate label for that component. The popup box will close and the annotated component will be highlighted with the color representing the chosen label. The label name is also assigned to the hidden input field corresponding to that component inside the form.


Annotation: Label selection - Micr...

Please select the appropriate label name:

- Title of article *
- Date/Time of article *
- Reporter Name
- Source Station *
- Country where news occurred *
- Image supporting contents of article *
- Link supporting contents of article
- Main content of article *
- Supporting content of article
- Subheaders
- Links to related articles
- Newsletter
- Site images
- Site content
- Site links/navigation
- Ads
- Search

* denotes higher priority labels

Gateway to report more financial details - Tech News - CNET.co...

Gateway to report more financial details

By Dawn Kawamoto and Ian Fried

Staff Writers, CNET News.com

January 24, 2002, 11:10 AM PT

Gateway is giving out new viewfinders to Wall Street.

The PC maker, which has been hit harder than its competitors in a brutal to give greater details of its financial data beginning with the release of i results Thursday, a move that analysts say will help them devise more a forecasts for the company.

Figure 2. When the user selects a label from the popup box (left), the component is highlighted with the corresponding color.

When the annotation is completed, the user submits the form and the components IDs are written to a result file together with the labels that the user tagged them with. Post-processing is done on the result file to match the labels to the original components via the IDs.

The annotation system can be found online at http://www-appn.comp.nus.edu.sg/~laihuiju/hyp.

4.2 Features

Besides the advantages of an automated system, our annotation system has many other features. When the user selects a label for a component, the component is highlighted, enabling the user to visually keep track of the components that have already been annotated.

Perl scripts were used for processing of output of the form web pages, as well as tasks like keeping track of the number of annotations for each web page. We have chosen a maximum of three annotations for each page in this project. Users are required to enter their email addresses when they enter the system and when they annotate a web page, so the system can keep track of the last page each user has annotated. When users enters the annotation system again, they will be redirected to the next page with less than three annotations and which they have not annotated.

4.3 Pre-processing of web pages

We chose to develop the annotation system to work as described in the previous section because we felt it was an intuitive method that the human user could easily understand. Javascript and CSS (Cascading Style Sheets) were used to implement the passing of the selected label name from the popup box back to the main browser window containing the form web page, as well as the highlighting of the component with the corresponding color.

To make it possible for the components to be clicked on and for the user's selected label to be stored in a hidden input field with the component ID, we have to insert two HTML code fragments around each component (one each before and after).

An example component with the code fragments around it is:

```
<span id=x1 onclick=change(this.id) class=parcels_none>
<b>Example text</b>
</span><input type=hidden name=x1name id=x1name value="">
```

where x1 and x1name are the IDs for this component and its hidden input field.

However since HTML is flexible and allows many types of files to be embedded in it (e.g. Javascript, Java applets, multimedia files), it also means HTML is complex and difficult to parse. As a result, this code insertion was the main difficulty encountered during pre-processing.

Our initial method treated the HTML source of the web pages as flat text. We simply went through the source line by line and replaced HTML opening and closing tags with the tag and our code fragments, incrementing the numerical component ID each time. However, this only worked if there was only one component on each line of the source, as we were unable to detect the boundaries of each component. Also, while this could work for a web page with components laid out linearly, it could not handle average web pages where components were nested. The code would be inserted multiple times for a deeply nested component (such as "<b><i><u>Example Text</u></i></b>").

Our next approach was to make use of the DOM tree's hierarchical structure. HTML Tidy was first used to clean up the source, correct any mistakes and convert the web page into a DOM tree. We were then able to insert the code fragments in the right places by traversing the tree recursively. To prevent the code from being inserted multiple times for nested components, it was only inserted when at the text nodes of the tree (the leaf nodes). The code was also inserted around image tags to allow images to be annotated. Also, since we only want to annotate components visible in the browser window, the code was not added around tags not located in the <body> of web pages and form-related components such as <select> or <textarea>.

Since we used a DOM tree representation of the web page, certain nodes and attributes could be modified or excluded when printing out the final processed page. Form and input nodes are not printed as they could affect the functioning of our annotation form. Links are all made blue in color and underlined, and their href and onclick attributes removed to prevent the user from being taken to another page while annotating. For images, since we are unable to include all the original images from the web pages, they are replaced with image placeholders. The images would have to be annotated based on their size and position relative to other components, and this is one of the limitations of the annotation system. Also, to prevent conflict with the system's Javascript and CSS, the text nodes of <script> and <style> tags as well as the src attribute of <script> tags are not printed. However, because Javascript can be inserted almost anywhere in a web page, it is difficult to remove all its elements completely.

The above solution is still incomplete. As mentioned and shown in an example given earlier in this section, one code fragment each has to be added before and after every component. However, while traversing the DOM tree, they are inserted separately. This results in a mismatch of the component IDs in the two code fragments, and an additional algorithm had to be implemented for matching the IDs.

4.4 Post-processing of result files

Each result file, which corresponds to a particular web page, contains a maximum of three sets of annotations by different users. It has to be post-processed to match the users' labels for each component with the component text itself. Compared to the pre-processing of web pages, it is a simpler task with two main steps. Firstly, for each component, one label has to be chosen. However, not all components may have been annotated by three users. If only one user has annotated a component, his or her label will be used. If two users have annotated a component, the two labels have to be the same (if not they are discarded). If three users have annotated a component, the majority is chosen (they are discarded if they are all different).

The next step will be extracting the text corresponding to each component and matching it with the labels chosen in the previous step. The labels and component text will then be written to a file, which can be used to build feature vectors for machine learning.

5. Testing

The data collected through the annotation system was used in testing the Text Detection module. Unfortunately, due to the lack of time, we were unable to collect as much data as we had hoped to. 21 web pages (from 12 news web sites) were annotated and 1271 text blocks were obtained from these pages.

The table on the right contains a breakdown of the text blocks by label. Surprisingly, no text blocks corresponding to Country where news occurred were annotated, thus we are unable to test and analyze the performance of our system for this label.

Label	**Count**
Title of article	22
Date/Time of article	23
Reporter Name	22
Source Station	5
Country where news occurred	0
Image supporting contents of article	25
Link supporting contents of article	44
Main content of article	201
Supporting content of article	81
Subheaders	19
Links to related articles	158
Newsletter	35
Site images	170
Site content	125
Site links/navigation	266
Ads	63
Search	12

Testing was performed on the Text Detection module alone (i.e. only lexical features), as well as on the entire PARCELS system (both lexical and stylistic features). For the former, the text blocks collected from the annotation system were given directly to the Text Detection module as input and 10-fold cross validation was carried out. Then the average accuracy with respect to the identification of each label was found.

To test the entire PARCELS system including the Stylistic Detection module, co-training was carried out. Co-training is carried out as follows: both machine learners are trained separately on a common training set, then used to classify data in a common testing set. The most confident classifications given by each learner are then considered as "correct" and added into the training set, and training/classification iterations can be performed for many rounds.

6. Results

The results of testing with respect to each PARCELS label are shown in the table below. “Baseline” denotes the accuracy of a baseline algorithm that relies solely on probability, while “Text Detection” denotes the average accuracy of 10-fold cross validation carried out on the Text Detection module alone, and “Co-training” denotes accuracy obtained from co-training rounds with the Stylistic Detection module.

Label	**Baseline**	**Text Detection**	**Co-training**
Title of article	0.017309205	0.181818182	0.227272727
Date/Time of article	0.018095987	0.434782609	0.47826087
Reporter Name	0.017309205	0.545454545	0.590909091
Source Station	0.00393391	0	0
Country where news occurred	-	-	-
Image supporting contents of article	0.019669552	0.12	0.16
Link supporting contents of article	0.034618411	0.022727273	0.068181818
Main content of article	0.158143194	0.835820896	0.910447761
Supporting content of article	0.063729347	0.222222222	0.24691358
Subheaders	0.014948859	0	0.105263158
Links to related articles	0.124311566	0.550632911	0.582278481
Newsletter	0.027537372	0.114285714	0.228571429
Site images	0.13375295	0.770588235	0.805882353
Site content	0.098347758	0.344	0.432
Site links/navigation	0.209284028	0.676691729	0.691729323
Ads	0.04956727	0.095238095	0.365079365
Search	0.009441385	0.583333333	0.666666667

Performance was highest for Main content of article as expected, and following that are Site images and Site links/navigation. The Text Detection module outperformed the baseline algorithm for almost all the labels, except for Source Station, Link supporting contents of article, and Subheaders. This is probably due to the fact that some of these are very similar to other labels, e.g. Subheaders may be similar to Title of article. Also, Link supporting contents of article may be difficult to identify based on lexical features alone, as it is distinguished from other types of links mainly by virtue of its position. Carrying out co-training also improved performance for almost all labels.

During training with BoosTexter, the most useful features were found to be *total-tags*, *total-words*, *top-keywords*, *Date*, *conjunctions*, *nouns* (generic), *text-links*, *anchors*, *contains-by*, *sentence-begins-with-verb*, *contains-newsletter* (specially designed).

7. Conclusions

7.1 Discussion

Although we had limited time during this project, we were able to complete a substantial amount of work with regards to implementing and integrating the PARCELS system. We have attempted to tag news article web page components with domain specific tags, as compared to general tags used during content extraction. We have also created our own features in addition to using generic lexical features, in order to tag components with labels specific to our domain. An annotation system was also built for annotation of web pages, which was difficult due to the complexity of HTML.

7.2 Recommendations for future work

As mentioned earlier in section 3.2, our identification task is not a multi-label one. However, it is highly likely in real life that more than one label may be assigned to one text block. For example, in the main content of a news article, there could be other components present, such as the source station, country, or even the date/time or reporter name. The identification task could be extended to a multi-label one to cover such cases. Instead of converting to a multi-label task, we could instead extract these components located within other components. This task is similar to term extraction, which can be done using machine learning as well (Turney, 2000).

We could also extend our identification task to other domains instead of concentrating on news article web pages only. The machine learning approach we used is easily adaptable as compared to hand-coded rules, and since the domain-specific PARCELS labels were not hard-coded into the form web pages during pre-processing, they can easily be replaced with new labels.

Since pre-processed form web pages in the annotation system may look very different from the original pages due to many factors (e.g. replacement of images with placeholders, removal of Javascript, CSS) thus causing a possible loss of accuracy during annotation, this can be countered by also displaying the original page to the user for comparison.

References

Cai, L. and Hofmann, T. (2003). Text Categorization by Boosting Automatically Extracted Concepts. In Proceedings of SIGIR-03, 26th ACM International Conference on Research and Development in Information Retrieval (Toronto, Canada, July 28 - August 1, 2003), ACM, New York, 2003, pp. 182-189.

DiPasquo, D. (1998). Using HTML Formatting to Aid in Natural Language Processing on the World Wide Web. Senior Honors Thesis, Carnegie Mellon University; 1998.

Evans, D.K. and Klavans, J.L. (2003). A Platform for Multilingual News Summarization. Under review for the Workshop on Multilingual Summarization of the Association for Computational Linguistics.

Gupta, S., Kaiser, G., Neistadt, D. and Grimm, P. (2003). DOM-based Content Extraction of HTML Documents. In Proceedings of 12th International World Wide Web Conference (Budapest, Hungary, May 20-24, 2003), pp. 207-214.

Lee, C.H. (2004). PARCELS: PARser for Content Extraction and Logical Structure (Stylistic Detection). Honours Year Final Report, National University of Singapore; 2004.

Mann, W. and Thompson, S. (1988). Rhetorical Structure Theory: Towards a Functional Theory of Text Organization. Computational Linguistics, Vol. 22, Issue 3, September 1996, pp. 409-419.

Porter, M.F. (1980). An algorithm for suffix stripping. Program, Vol. 14, No. 3, July 1980, pp. 130-137.

Schapire, R.E. and Singer, Y. (2000). BoosTexter: A Boosting-based System for Text Categorization. Machine Learning, Vol. 39, No. 2/3, May/June 2000, pp. 135-168.

Soderland, S., Fisher, D. and Lehnert, W. (1997). Automatically Learned vs. Hand-crafted Text Analysis Rules. Technical Report TE-44, National Centre for Intelligent Information Retrieval (CIIR), University of Massachusetts, Amherst MA, 1997.

Turney, P.D. (2000). Learning Algorithms for Keyphrase Extraction. Information Retrieval, Vol. 2, No 4, 2000, pp. 303-336.

Vivaldi, J., Màrquez, L. and Rodríguez, H. (2001). Improving Term Extraction by System Combination using Boosting. In Proceedings of the joint ECML-PKDD'01 Conference. Freiburg, Germany, 2001.

Yang, Y. (1997). An Evaluation of Statistical Approaches to Text Categorization. Information Retrieval, Vol. 1, No. 1-2, 1999, pp. 69-90.

Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99), pp. 42-49.

Yu, S., Cai, D., Wen, J.R. and Ma, W.Y. (2003). Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation. In Proceedings of 12th International World Wide Web Conference (Budapest, Hungary, May 20-24, 2003), pp. 11-18.