

Adaptive Web-page Content Identification

John Gibson
+01 781-271-4757
jgibson@mitre.org

Ben Wellner
+01 781-271-7191
wellner@mitre.org

Susan Lubar
+01 781-271-2860
slubar@mitre.org

The MITRE Corporation
202 Burlington Rd.
Bedford, MA 01730-1420 USA

ABSTRACT

Identifying which parts of a Web-page contain target content (e.g., the portion of an online news page that contains the actual article) is a significant problem that must be addressed for many Web-based applications. Most approaches to this problem involve crafting hand-tailored rules or scripts to extract the content, customized separately for particular Web sites. Besides requiring considerable time and effort to implement, hand-built extraction routines are brittle: they fail to properly extract content in some cases and break when the structure of a site's Web-pages changes. In this work we treat the problem of identifying content as a sequence labeling problem, a common problem structure in machine learning and natural language processing. Using a Conditional Random Field sequence labeling model, we correctly identify the content portion of web-pages anywhere from 80-97% of the time depending on experimental factors such as ensuring the absence of duplicate documents and application of the model against unseen sources.

Categories and Subject Descriptors

I.2.6, I.2.7 [Artificial Intelligence]: Natural Language Processing – text analysis, Learning – Parameter Learning.

General Terms

Algorithms, Experimentation.

Keywords

Conditional random fields, content identification, maximum entropy markov models, sequence labeling.

1. INTRODUCTION

Web pages containing news stories also include many other pieces of extraneous information such as navigation bars, JavaScript, images and advertisements. There are a number of tasks that necessitate the extraction of just the news article from these pages. This might be done to provide input into a database or into an application such as a Natural Language tool, index for a search engine or duplicate detection tool. Another cause to extract just the news story is to re-display it on a small screen

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WIDM '07, November 9th, Lisbon, Portugal

Copyright 2007 ACM \$5.00.

Approved for public release; Distribution Unlimited; Case #07-0958


such as a cell phone or PDA. An example of identifying the embedded news article can be seen in Figure 1.

Typically, content extraction is done via a hand-crafted tool targeted to handle a single web page format. This approach is brittle in that when the page format changes, the extractor is likely to break. Additionally, it is labor intensive since a new extractor must be written to handle each unique page format. In our experience, web page formats change fairly quickly and custom extractors often become obsolete a short time after they are written. Further, some websites use multiple formats concurrently and identifying each one and handling them properly makes this a complex task. As part of a larger project, we initially developed such site-specific content extractors and found them to be unworkable as a long-term solution due to the aforementioned problems.

The approach described in this paper is meant to overcome these issues. The data set for this work consisted of web pages from 27 different news sites. The sites are visually similar in that they contain a news article surrounded by other information. However, the underlying HTML for creating this layout varies amongst the sites. For example, while some sites separate sections of the article content with paragraph tags, others segment the sections with tags such as `<div>`, `
`, or `<table>`.

Identifying portions of relevant content in web-pages can be construed as a sequence labeling problem – i.e. each document is broken into a sequence of blocks and the task is to label each block as Content or NotContent. We employed three different statistical machine learning methods to learn how to label sequences of blocks so as to identify the content. Our best system, based on Conditional Random Fields, can perfectly identify the content portions from new, unseen web-pages *at unseen web-sites* over 80% of the time¹. Under less strict conditions (e.g. allowing duplicate articles and/or articles from a seen web-site in the test data), the document level accuracy improves to nearly 98%. In all cases, the system identifies Content blocks with recall above 99.5% and precision above 97.9%. These results support the claim that machine learning-based content extraction is a very viable alternative to hand-crafted patterns.

¹ A demo of this system is available at <http://cedarpub.mitre.org/>

Figure 1.  Indicates news story content



2. RELATED WORK

Web-page Content Identification

Information Extraction

Information Extraction is the task of identifying “nuggets” of information such as dates, authors, or prices, from structured or unstructured text. As described by Laender, et al [1], the most common technique used for information extraction involves constructing site-specific programs called wrappers. This approach possesses the characteristics described above – wrappers are generally brittle and labor intensive. There have been numerous attempts to address these problems using various techniques. An explanation and examples of the automatic learning of rules, called wrapper induction can be found in work by N. Kushmerick [2] and Muslea et al [3]. An approach to automatically adapting to page format changes is described in [4]. Laender et al[1] provide an overview of information extraction tools which use wrappers to process Web pages. Wrapper induction is related to our task in that it is identifying specific content in text; however, the information being extracted is generally more structured than the free-text news articles that we are identifying. Therefore, items identified via wrapper induction are more easily recognizable using patterns.

Content Extraction

Content Extraction tools such as Columbia University’s Crunch [5] aim to reduce the size of web pages by removing what they deem as noise or clutter from the pages. Additionally, a tool by the Document Analysis and Recognition Team (DART) at BCL Computers Inc. [6] further reduces text by providing a summary of what remains. These tools are motivated by a variety of goals including paring down pages for the visually impaired [5], producing lighter weight content for small screen devices such as PDAs [5, 6] and reducing page complexity for subsequent processing as in MetaNews [7] and the forthcoming CLEANVAL challenge task as part of the Web as a Corpus Workshop [8].

Content Extraction tools are related to our work in that they are focused on determining which part(s) of web pages are relevant to their goal. In general, they are addressing a broader problem than what this paper addresses. Content Extraction tools are meant to take any type of web page as input, while we are focused solely on pages containing news articles. In addition, while the Content Extraction tools aim to reduce page size, they are not geared toward identifying a specific portion of the page to keep, as we are with news articles.

News Story Identification

The Columbia Newsblaster project [9] and MetaNews [6] both concentrate on gathering news articles on the web. MetaNews uses a two-phased approach. First, it carries out noise removal by throwing out HTML tags that it believes will not contain content. Next it uses pattern matching on the reduced page to extract news articles. Patterns for MetaNews are manually defined for each news site, and no automatic learning is involved. Thus although pattern writing is simpler than for traditional wrapper approaches, this tool is still likely to fail if a page format changes, and adding new sites requires some manual labor.

The Columbia Newsblaster team originally used individual site wrappers to identify news articles. They determined that this approach was difficult for handling new sites. As a result, they implemented a machine learning based approach which is similar to ours. The module relies on “simple surface characteristics of the text” to classify blocks of text as part of an article, or into various other categories such as title, caption, or other.

Sequence Labeling

Sequence labeling methods have seen widespread use in the natural language processing community for tasks such as part-of-speech tagging [9], noun-phrase chunking [10] and other related tasks. For such problems, the elements in the sequence are words and a sequence typically consists of a sentence.

Sequence labeling methods have been applied to sequences where the elements are units other than single words, however. Examples include labeling sequences of clauses [11] or labeling argument constituents of verbs [12]. Most related to our work from a sequence labeling perspective, is work in extracting tables from ASCII files [13]. Here, sequences were entire documents with each line in the file being an element. The task was to extract sequences of lines that constituted a table. In our setting, the sequence elements are HTML blocks rather than lines in an ASCII file. The tasks are similar, however, in that they both consider sequence elements containing multiple words with potentially complicated internal structure.

3. CONTENT IDENTIFICATION AS SEQUENCE LABELING

The problem we have addressed in this work is that of identifying the portions of news-source web-pages that contain relevant content – i.e. the news article itself. There appear to be two general ways to approach this problem:

Boundary Detection Method. This approach involves identifying positions where content begins and ends, thus deriving a segmentation of the original Web-page delineating the content by assuming all parts of the page between a begin content marker

and an end content marker are content. The identification method could be any decision mechanism; we generally assume this to be a statistical classifier.

Sequence Labeling Method. The second approach is to divide the original Web document into a sequence of some appropriately sized units or blocks and to then categorize each block as Content or NotContent. Given this formulation as a sequence labeling problem, many robust statistical methods developed for labeling sequences (in the Natural Language Processing community and elsewhere) are applicable.

In this work, we focus largely on the sequence labeling method. This is motivated by a number of factors. First, a number of web pages contain ‘noncontiguous’ content. By this we mean that the paragraphs of the article body have other page content, such as advertisements, interspersed. Sequence labeling methods seem preferable in such situations since the boundary detection methods aren’t able to nicely model transitions from Content to NotContent back to Content, for example. Another problem with the boundary detection method is that if a boundary is not identified *at all* (e.g. the beginning of content boundary), an entire section of content can be missed. There are various heuristic methods one could apply to ensure this doesn’t happen, but none would be well-principled. Finally, when developing a statistical classifier to identify boundaries, there are many, many more negative examples of boundaries than positive ones. Of course, it may be possible to sub-sample negative examples or identify some reasonable set of candidate boundaries and train a classifier on those, but this complicates matters greatly.

While our efforts here focus on the sequence labeling approach, we did perform some initial experiments using the boundary detection method which involved identifying the beginning and ending of content using a statistical classifier. We found the results to perform no better than the lowest performing sequence labeling approach. This, after spending considerable effort to post-process the statistical classifier output to derive the most sensible content boundaries. For example, if no begin content boundary was identified by the classifier (i.e. it had a “yes” probability > 0.5) we selected the candidate begin content boundary that had the highest classifier confidence.

4. MODELS

This section describes the three statistical sequence labeling models employed in our experiments: Conditional Random Fields (CRF), Maximum Entropy Classifiers (MaxEnt), and Maximum Entropy Markov Models (MEMM).

Conditional Random Fields

Conditional Random Fields (CRFs) [14] are probabilistic models that produce a conditional distribution over a set of output (i.e., predicted) variables when provided with fixed values to a set of input (i.e., given) variables. An important special case of CRFs are *sequence CRFs* in which the output variables are arranged in a linear-chain in which a first-order Markov assumption holds: an output variable is only dependent on the variables adjacent to it (i.e., immediately to the right and left) in the sequence.

Let $\mathbf{x} = x_1, x_2, \dots, x_n$ be a sequence of observations such as a sequence of words, paragraphs, or, as in our setting, a sequence of HTML “segments”. Given a set of possible output values (i.e.,

labels), sequence CRFs define the conditional probability of a label sequence $\mathbf{y} = y_1, y_2, \dots, y_n$ as:

$$p(\mathbf{y} | \mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \exp\left(\sum_{i=1}^n \sum_k \lambda_k f_k(y_i, y_{i-1}, x, i)\right)$$

where $Z_{\mathbf{x}}$ is a normalization term overall possible label sequences. The feature functions f_k are arbitrary functions over the current and previous labels, y_i and y_{i-1} , the entire observation sequence \mathbf{x} and the current position, i . Often, the range of such functions is $\{0,1\}$, though the range may extend to arbitrary real-values, generally. For example, a feature function may generally have value 0 for most inputs, but have a value of 1 when $y_i = \text{Content}$, $y_{i-1} = \text{NotContent}$ and x_i contains the word “TITLE”. Associated with each feature function, f_k is a learned parameter λ_k that captures the strength and polarity of the correlation between the label transition, y_{i-1} to y_i , and the predicate over the observation sequence. Thus, the above feature with a high positive weight would indicate that the word “TITLE” appearing in a segment, x_i , is correlated with the current segment labeled as Content and the previous segment labeled as NotContent.

Note also that feature functions in practice often ignore the previous label, and simply correlate the predicate over the observations with the label at the current position (not the current *and* previous position). Such features are sometimes called *node features*, while features that do pay attention to the current and previous position are termed *edge features*.

Training

Assume a set of training data consisting of a set of pairs of sequences – the first element of each pair being a label sequence and the second being a corresponding observation sequence: $D = \{(y^{(1)}, x^{(1)}), (y^{(2)}, x^{(2)}), \dots, (y^{(m)}, x^{(m)})\}$. The model parameters are learned by maximizing the conditional log-likelihood of the training data which is simply the sum of the log-probabilities assigned by the model to each label-observation sequence pair:

$$L(D) = \sum_{i=1}^m \log(p(y^{(i)} | x^{(i)})) - \sum_k \frac{\lambda_k}{2\sigma^2}$$

The second term in the above equation is a Gaussian prior over the parameters, which helps the model to overcome over-fitting. Lower variance values have the effect of proportionally penalizing the log-likelihood when feature values stray far from 0. While the log-likelihood surface is convex (i.e., there is a single global maximum), maximizing the above expression over a set of training data isn’t possible in closed form, and general purpose convex optimization routines are used in practice. The optimization process requires evaluating the log-likelihood and its gradient for the current set of parameters at each iteration. Full inference over the model must be performed to derive the distribution over all possible labelings. This distribution is needed to compute the feature expectations required for the gradient term and to compute the normalization term, $Z_{\mathbf{x}}$. Inference is performed using a variant of the forward-backward algorithm[14]. As this must be performed on each iteration, CRF training is relatively compute-intensive as compared with the methods below.

Decoding

Given a trained model, the problem of decoding in sequence CRFs involves finding the most likely label sequence for a given observation sequence. The number of possible label sequences is very large: there are N^M possible label sequences where N is the number of labels and M is the length of the sequence. Dynamic programming, specifically a variation on the Viterbi algorithm [14], can find the optimal sequence in time linear in the length of the sequence and quadratic in the number of possible labels.

Maximum Entropy Classifier

Maximum Entropy (MaxEnt) classifiers have been used with great success in areas such as text processing, natural language processing (e.g. parsing) and many other areas. Briefly, MaxEnt classifiers are conditional models that given a set of parameters produce a conditional multinomial distribution according to:

$$p(y = y_i | x) = \frac{\exp(\sum_k \lambda_k f_k(y_i, x))}{\sum_{y_j} \exp(\sum_k \lambda_k f_k(y_j, x))}$$

where y is a random variable over the possible outcomes or classifications (e.g. Content or NotContent), x describes the observed data (e.g. HTML pages), the functions f_k are features over the observed data and a particular outcome, and the λ_k are the weights, one associated with each feature. MaxEnt models tend to perform competitively with other discriminative learning methods such as support vector machines (SVMs) and benefit from their simplicity, robustness and scalability. In contrast with CRFs, MaxEnt models (and classifiers generally) are “state-less” and do not model any dependence between different positions in the sequence.

Maximum Entropy Markov Models

A final set of models we consider in this work are Maximum Entropy Markov Models (MEMMs). MEMMs model a state sequence just as CRFs do, but use a “local” training method rather than performing global inference over the sequence at each training iteration. Specifically, a model is learned to classify each block in the sequence using features of the observation sequence as well as the label of the previous block according to the gold-standard training data. Viterbi decoding is employed as with CRFs to find the best label sequence. The model takes the form:

$$p(y | x) = \sum_i p(y_i | y_{i-1}, x)$$

$$\text{where } p(y_i | y_{i-1}, x) = \frac{\exp(\sum_k \lambda_k f_k(y_i, y_{i-1}, x, i))}{\sum \exp(\sum_k \lambda_k f_k(y_j, y_{i-1}, x, i))}$$

Because MEMMs use the gold-standard previous labels during training, and because they do not normalize over the entire sequence like CRFs, they are prone to various biases (e.g. the *label bias* [14]) that can reduce their accuracy.

5. DATA

Harvesting and Annotation

We used RSS feeds from Reuters and AP to acquire a list of article titles. Each title was submitted in a query to Google News, and the top ten results were downloaded. Because of the large variance in the pagination methods of each site we limited downloads to the first page of each article. The overall goal of our project is to detect duplicate articles, and this method results in numerous duplicate articles. However, the large number of duplicate and near duplicate documents introduce bias into our training set.

We analyzed and compensated for the bias in section 6 (Experimental Setup, Data Set Creation). In total we harvested 2577 documents from 49 separate websites.

The content of each document was annotated with a combination of site-specific wrappers and manual tagging. Although we harvested data from 49 separate websites, we annotated a total of 27 because some sites were too difficult to tag automatically and others did not yield enough documents to justify writing a script to automatically tag them. As mentioned, the process of developing hand-crafted, site-specific extractors was difficult and time-consuming, however, it proved to be an preferable alternative to full annotation. For example, our annotation guidelines changed along the way, and we were able to realize these changes in the data by simply adjusting the site-specific wrappers.

We developed guidelines to make the data uniform across different sites. Many stories would end with content that was optional from site to site. We excluded content such as “additional reporting by...”, article termination delimiters like “--”, and “On the Net” lists of links. If these text ranges were included as core content, documents that were in fact identical articles would instead be flagged as near duplicates. Similarly some documents would include photos with captions, often in different places with respect to the rest of content. These were excluded as well to preserve accurate duplicate detection.

As part of our data preparation, we also annotated pairs of documents that we considered identical or near duplicates due to the similarity of their content. In the end we annotated 1620 documents from 27 different sources. Duplicate analysis revealed that there were 388 distinct articles.

Dividing into Blocks

As a first step, we sanitize the raw HTML and transform it into XHTML using a combination of Tagsoup² and BeautifulSoup³. In the rare case that a document that can not be transformed into XHTML, we discard it. Otherwise we tokenize the document, wrapping each word in a `<lex>` tag. We exclude all words inside style and script tags because it is safe to assume that they will never contain any content or metadata. Next we create blocks of tokens by wrapping sequences of `<lex>` tags with `` tags. Blocks are bounded by any tag except the

² Tagsoup, by John Cowan: <http://ccil.org/~cowan/XML/tagsoup/>

³ BeautifulSoup, by Leonard Richardson and others: <http://www.crummy.com/software/BeautifulSoup/>

following: `<a>`, `<ins>`, ``, ``, `<bdo>`, ``, ``, `<dfn>`, `<code>`, `<samp>`, `<kbd>`, `<var>`, `<cite>`, `<abbr>`, `<acronym>`, `<q>`, `<sub>`, `<sup>`, `<tt>`, `<i>`, ``, `<big>`, `<small>`, `<u>`, `<s>`, `<strike>`, `<basefont>`, and ``. Each block contains at least one `<lex>` tag and is wrapped as tightly as possible with the `` tag. In rare cases a block must consist of more than one `` tag because of the structure of the original HTML.

Although the words in the `<title>` are tokenized, those tokens are not added to any block. Instead, they are used to generate features for other blocks (see Title Matching, section 6).

Finally, we create features from the `` tags in each block.

There is a large skew of NotContent vs. Content blocks. In all of our training data we had 234,436 NotContent blocks and only 24,388 Content blocks. This 10:1 ratio was also present in our individual data sets.

6. EXPERIMENTAL SETUP

Feature Selection

We generated 11 different feature types for classification.

Words. A case-insensitive bag-of-words count of the tokens in a block.

Inverse Stop-Wording. Instead of using all of the tokens in a block for the bag-of-words count, this feature only uses the tokens identified as stop words. This feature was intended to help identify that English prose was being used in a block without becoming too dependent upon the particular vocabulary of an article. Obviously this feature type is mutually exclusive with the normal bag-of-words feature.

Named Entities. A count of the named entities in a block, grouped by the following types: person, organization, location, date, time, monies, and percentages. Furthermore tokens that are identified as part of a named entity are not included in the bag-of-words feature, with the following exceptions: Reuters, AP, A.P., and Associated Press.

Title Casing. Features are generated when every token in a block begins with an upper case letter or when any token begins with a lower case letter.

Anchor Percentage. The percentage of tokens in a block contained within an anchor tag (`<a>`).

Title Matching. If two or more tokens in a block, in order, match a subsequence of the tokens contained in the `<title>` of the document then the percentage of the title that matched is recorded. This feature is intended to aid in identifying the title of the document in later passes, however it was included in content classification to simplify processing.

Ancestor Tags. The names of the parent and grandparent (if present) tags of a block.

Descendant Tags. The names any descendant tags found within a block. `<a href>` are counted separately from `<a name>` tags. `` are equivalent to `` and `<i>` are equivalent to ``.

Sibling Tags. The names of the previous and next sibling tags of the current block (if present).

Word Count. A count of the tokens found in a block.

After Image Tag. A feature that indicates if an `` appears before the current block and after the previous one. This feature is intended to exclude photo captions from inclusion in the content.

We evaluated the value of the features by turning them off one-by-one and testing the results with our best classifier.

Data Set Creation

To avoid bias in our data we used four separate data splitting strategies to evaluate our performance. Due to the nature of our data harvest we had numerous exact and near duplicate documents in our data set. Of the 1620 documents, only 388 were distinct articles. When the same article appears in both the training and testing set, it could distort the value of the word features. The second source of potential bias arises from the websites themselves. If documents from the same website appear in both the training and testing set then the idiosyncrasies of the website can influence the classifier.

We ran four separate cross validation experiments to measure the bias introduced by duplicate articles and mixed sources.

Duplicates, Mixed Sources. This was a simple split of the documents with 75% in the training set and 25% in the testing set.

No Duplicates, Mixed Sources. This split prevented duplicate documents from spanning the training and testing boundary. The documents were grouped by article and then all of the documents in an article were assigned to the testing set until at least 25% of the documents were in the testing set.

Duplicates Allowed, Separate Sources. In our data, the number of documents from each source varies from as few as 6 to as many as 250. This variance prevented a simple splitting of the sources. Instead we created four separate bundles each containing 6 or 7 sources. We filled the bundles in round-robin fashion by selecting a source at random and weighed the probability of selecting a source by the number of documents that it contained. This kept the bundles at a roughly equal size. Three bundles were assigned to training and one to testing.

No Duplicates, Separate Sources. To eliminate both duplicate and source bias, we bundled the sources as in duplicates allowed, separate sources. Then the articles were grouped by training and testing sets, and if any overlapped, half of the overlapping articles were removed from the training set and the other half from the testing set to ensure that no duplicates crossed the training/test set boundary. If it wasn't an even split of articles then testing was favored over training.

7. RESULTS AND ANALYSIS

We carried out a variety of experiments using the data splits and methodology described in Section 6.2. We examined the performance of the different statistical methods described earlier with different Gaussian prior parameters and different sets of features.

Model Options

A major focus of our experiments involved looking at how different feature sets performed. Each of the three statistical models we have employed (CRF, MEMM, MaxEnt) include different options in how certain types of features are constructed. Specifically, the CRF and MEMM models include two binary options for automatically expanding model features.

The first option, when “on” indicates that *edge features* should be created out of all predicates over the observation sequence as well as the default node features.

The second option introduces ‘history predicates’. For a position, i , in a sequence, all the predicates at position $i-1$ are added as new predicates at position i . For example, if the predicate ContainsWord:title was present at position $i-1$ the predicate PREV-ContainsWord:title would be introduced at position i indicating that the word “title” was present in the previous block.

All three of the models include a Gaussian prior parameter. We adjusted the values for the parameter to see its affect on performance.

Table 1. Document level accuracy (percentage of documents correctly labeled) and document level precision and recall with respect to Content labels for each classifier over the 4 data setups using all the available features. Model options include the Gaussian prior (σ) and the edge (E) and history (H) feature options.

Classifier	Doc. Acc.	Doc. Prec.	Doc. Recall	Block Prec.	Block Recall
No Duplicates, Separate Sources					
CRF $\sigma=100$	0.804	0.839	0.933	0.979	0.995
MEMM $\sigma=100,H$	0.777	0.862	0.866	0.986	0.969
MaxEnt $\sigma=10,H$	0.576	0.657	0.750	0.960	0.981
Duplicates Allowed, Separate Sources					
CRF $\sigma=1$	0.876	0.907	0.945	0.992	0.995
MEMM $\sigma=1,H$	0.831	0.889	0.902	0.990	0.977
MaxEnt $\sigma=10,H$	0.638	0.707	0.787	0.950	0.983
No Duplicates, Mixed Sources					
CRF $\sigma=1, E$	0.961	0.979	0.978	0.997	0.998
MEMM $\sigma=1,H,E$	0.912	0.948	0.951	0.994	0.996
MaxEnt $\sigma=100,H$	0.779	0.857	0.883	0.987	0.992
Duplicates Allowed, Mixed Sources					
CRF $\sigma=10$	0.977	0.985	0.989	0.999	0.999
MEMM $\sigma=1,H,E$	0.945	0.963	0.980	0.997	0.998
MaxEnt $\sigma=100,H$	0.920	0.955	0.959	0.996	0.996

Table 2. Comparative feature analysis for the best system (CRF $\sigma=100$) with No Duplicates; Separate Sources.

Feature Set	Doc. Acc.	Doc. Prec.	Doc. Recall	Block Prec.	Block Recall
All Features	0.804	0.839	0.933	0.979	0.995
No descendant tags	0.809	0.847	0.933	0.979	0.995
No named entities	0.803	0.860	0.907	0.985	0.992
No title matching	0.794	0.825	0.939	0.979	0.996
No anchor percent	0.793	0.833	0.927	0.977	0.995
No word count	0.789	0.830	0.918	0.978	0.994
No sibling tags	0.782	0.809	0.940	0.975	0.996
No after img	0.764	0.804	0.906	0.977	0.993
No title case	0.762	0.815	0.899	0.975	0.989
Inverse Stop Words	0.701	0.798	0.820	0.963	0.971
No ancestor tags	0.691	0.715	0.927	0.967	0.996
No words	0.636	0.749	0.733	0.974	0.914

Results and Analysis

Evaluation Metrics

We evaluated all of our results at the block level and at the document level. Block Precision is the ratio of the number of correctly labeled Content blocks to the number of proposed Content blocks. Block Recall is the ratio of the number of correctly labeled Content blocks to the number of true Content blocks according to the gold-standard. Document Precision is the percentage of *documents* with 100% Block Precision and Document Recall is the percentage of documents with 100% Block Recall. Document Accuracy describes the percentage of documents that were correctly labeled (i.e., all blocks were labeled correctly).

Results summary

Our main results are summarized in Table 1. Each of the three different models, CRF, MEMM and MaxEnt are evaluated on each of the four different data setups. All results are aggregated from four-fold cross validation. As made clear in the table, in all cases the CRF performs markedly better than the other approaches. MaxEnt, with no ability to capture sequential nature of the problem results in significantly lower performance. The MEMM results generally lie right in between.

Feature Type Analysis

Table 2 shows results for the CRF when individually removing each class of feature (described in 6.2) using the “No Duplicates; Separate Sources” data setup. In most cases, removing a feature class results in a small drop in performance (or a small gain in a couple cases), but even considering that the test documents are drawn from different sources and that duplicates do not appear in the test set, the CRF performs substantially worse when removing the words as features (“No words”). This is somewhat surprising and indicates that a lexicalized model is important. That is, considering only the document structure is insufficient. The fact that these individual word features are so important indicates the

problem may be more closely related to text classification and topic modeling than one might initially suspect.

Contiguous vs. Noncontiguous Performance

Another performance factor that we measured was contiguous vs. noncontiguous performance. Although in many documents the core content is an unbroken series of blocks, in many others the flow of Content blocks is interrupted by advertisements, site navigation, or other NonContent blocks. Table 3 shows the document level results of Table 1 broken out by contiguousness.

Table 3. Contiguous and noncontiguous Document level accuracy (percentage of documents correctly labeled) and document level precision and recall with respect to Content labels for each classifier over the 4 data setups using all the available features. Model options include the Gaussian prior (σ) and the edge (E) and history (H) feature options.

Classifier	Non-Contiguous			Contiguous		
	Acc.	Prec.	Rec.	Acc.	Prec.	Rec.
No Duplicates, Separate Sources (NonContig=243, Contig=680)						
CRF $\sigma=100$	0.597	0.609	0.935	0.878	0.924	0.933
MEMM $\sigma=100,H$	0.584	0.604	0.910	0.846	0.963	0.856
MaxEnt $\sigma=10,H$	0.342	0.346	0.806	0.660	0.788	0.741
Duplicates Allowed, Separate Sources (NonContig=417, Contig=1203)						
CRF $\sigma=1$	0.746	0.766	0.940	0.921	0.957	0.946
MEMM $\sigma=1,H$	0.652	0.687	0.901	0.894	0.961	0.903
MaxEnt $\sigma=10,H$	0.307	0.311	0.699	0.752	0.861	0.802
No Duplicates, Mixed Sources (NonContig=423, Contig=1210)						
CRF $\sigma=1, E$	0.943	0.961	0.973	0.968	0.986	0.980
MEMM $\sigma=1,H,E$	0.943	0.955	0.978	0.901	0.946	0.941
MaxEnt $\sigma=100,H$	0.745	0.836	0.858	0.791	0.864	0.892
Duplicates Allowed, Mixed Sources (NonContig=431, Contig=1189)						
CRF $\sigma=10$	0.944	0.964	0.974	0.988	0.992	0.995
MEMM $\sigma=1,H,E$	0.942	0.946	0.993	0.946	0.969	0.975
MaxEnt $\sigma=100,H$	0.893	0.937	0.946	0.929	0.961	0.963

Amount of Training Data

A final question we were interested in addressing in our experiments was the relationship between the amount of training data and performance. Table 4 shows the results for the CRF on the “No duplicates; Separate Sources” data setup with the full data set, with 2/3 of the data and with 1/3. The results here were obtained by training on three choices of two partitions and 1 partition within each training fold and averaging them. The results indicate a linear growth in performance as the amount of training data increases.

Error Analysis

Error analysis revealed a number of problems with NotContent blocks found interspersed within portions of Content being falsely labeled as Content. This pattern can be seen from Table 3

where Document recall is generally higher than precision. As an example, image captions often appear in the middle or adjacent to the article content and are sometimes incorrectly labeled as Content. As the image captions are well-formed English prose this confusion is understandable. On the recall errors side, we noticed that sometimes section headers would be incorrectly singled out as NotContent. Another category of recall errors involved the beginning and end of article boundaries where sometimes the first or last block would incorrectly be labeled NotContent.

We also noted a number of errors due to problems with the pre-processor. For example, we noticed blocks of Javascript included as document text which sometimes were identified as Content due to long runs of prose inside the Javascript. These pieces of Javascript should have been removed during pre-processing, but escaped our filtering mechanism.

Finally, it’s worth noting that given that the block-level Content precision and recall scores are quite high, it is clear that most documents contain minimal numbers of errors. These errors reduce the document level scores significantly, though many of these documents are in fact very close to perfect.

Table 4. Results with different amounts of training data using the CRF on the “No Duplicates; Separate Sources” setup.

Training Data Amount	Doc. Acc.	Doc. Prec.	Doc. Recall	Block Prec.	Block Recall
100%	0.804	0.839	0.933	0.979	0.995
67%	0.718	0.753	0.910	0.973	0.991
33%	0.681	0.714	0.910	0.974	0.991

8. CONCLUSION AND FUTURE WORK

Our results show that using machine learning techniques to identify the content of a news article on the web is a feasible alternative to site-specific wrappers. The performance on known websites is competitive with hard-coded wrappers, but the real value of the machine learning approach is the flexibility that it provides. The handcrafted method of extracting content is brittle at best over the long term. Handling website format changes and adding additional sites is a labor intensive endeavor that is out of reach for the majority end users. Although the document-level accuracy is only 80% with our best model under the strictest experimental conditions, it has been sufficient for our duplicate detection project. Moreover, the precision and recall for identifying Content blocks is very high. This indicates that even for documents containing errors, the vast majority of the blocks are correctly identified. Finally, our experiments indicate that it is likely that the performance would continue to increase with additional training data.

With regards to algorithmic performance, sequence labeling emerged as the clear winner with CRF edging out MEMM. Traditional maximum entropy classification was only competitive on the easiest of the four data sets and is not a viable alternative to site-specific wrappers.

Future work includes applying the techniques to additional data, including additional sources, different languages, and other types of data such as weblogs. Another interesting avenue to explore is the performance of semi-Markov CRFs[15] which, rather than labeling individual elements in a sequence, explicitly consider all possible segmentations (with segments up to a specified maximum length) within the model. This would allow for features over (hypothesized) *segments* of blocks (as well as label dependences between such segments) that might better capture lexical or structural differences between Content and NonContent sections.

Acknowledgements

This work was supported by the Department of the Army's Communications-Electronics Lifecycle Management Command (C-E LCMC) and performed under MITRE Mission Oriented Investigation and Experimentation (MOIE) Project M130 of contract W15P7T-07-C-F600, managed by Special Operations Command, Pacific. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the sponsor.

References

1. Laender, A., et al., *A Brief Survey of Web Data Extraction Tools*. SIGMOD, 2002. **31**(2).
2. Kushmerick, N. *Wrapper Induction: Efficiency and Expressiveness*. in *AAAI-98 Workshop on AI and Information Integration*. 1998.
3. Muslea, I., S. Minton, and C. Knoblock. *A Heirarchical Approach to Wrapper Induction*. in *Proceedings of the Third International Conference on Autonomous Agents (Agents '99)*. 1999. Seattle, WA.
4. Knoblock, C., et al., *Accurately and Reliably Extracting Data from the Web: A Machine Learning Approach*. Data Engineering Bulletin, 2000. **23**(4).
5. Gupta, S., et al., *Automating Content Extraction of HTML Documents*. World Wide Web - Internet and Information Systems, 2005. **8**(2): p. 179-224.
6. Rahman, A.F.R., H. Alam, and R. Hartono. *Understanding the Flow of Content in Summarizing HTML Documents*. in *International Workshop on Document Layout Interpretation and its Applications (DLIA)*. 2001.
7. Kang, D. and J. Choi. *MetaNews: An Information Agent for Gathering News Articles on the Web*. in *International Symposium on Methodologies for Intelligent Systems*. 2003.
8. WAC 2007, W.a.a.C. in *Web as a Corpus*. 2007. UCLouvain, Louvain-la-Neuve, Belgium.
9. Wellner, B. and M. Vilain. *Leveraging Machine-Readable Dictionaries in Discriminative Sequence Models*. in *Language Resources and Evaluation Conference (LREC 2006)*. 2006. Genoa, Italy.
10. Sha, F. and F. Pereira. *Shallow parsing with conditional random fields*. in *Proceedings of HLT-NAACL 2003*. 2003.
11. Palmer, A., et al. *A Sequencing Model for Situation Entity Classification*. in *Association for Computation Linguistics*. 2007. Prague, Czech Republic.
12. Marquez, L., et al. *Semantic Role Labeling as Sequential Tagging*. in *Conference on Natural Language Learning (CoNLL)*. 2005. Ann Arbor, MI.
13. Pinto, D., et al. *Table Extraction using Conditional Random Fields*. in *Proceedings of the ACM SIGIR*. 2003.
14. Lafferty, J.D., A. McCallum, and F.C.N. Pereira. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequences*. in *ICML 01: Proceedings of the Eighteenth International Conference on Machine Learning*. 2001. San Francisco CA, USA: Morgan Kaufmann.
15. Sarawagi, S. and W. Cohen. *Semi-Markov Conditional Random Fields for Information Extraction*. in *Proceedings of Neural Information Processing Systems*. 2004.