Content Extraction from News Pages Using Particle Swarm Optimization on Linguistic and Structural Features

Cai-Nicolas Ziegler Michal Skubacz

Siemens AG, Corporate Research & Technologies (IC 1)
Otto-Hahn-Ring 6, D-81730 München

{cai.ziegler, michal.skubacz}@siemens.com

Abstract

Today's Web pages are commonly made up of more than merely one cohesive block of information. For instance, news pages from popular media channels such as Financial Times or Washington Post consist of no more than 30%-50% of textual news, next to advertisements, link lists to related articles, disclaimer information, and so forth. However, for many search-oriented applications such as the detection of relevant pages for an in-focus topic, dissecting the actual textual content from surrounding page clutter is an essential task, so as to maintain appropriate levels of document retrieval accuracy. We present a novel approach that extracts real content from news Web pages in an unsupervised fashion. Our method is based on distilling linguistic and structural features from text blocks in HTML pages, having a Particle Swarm Optimizer (PSO) learn feature thresholds for optimal classification performance. Empirical evaluations and benchmarks show that our approach works very well when applied to several hundreds of news pages from popular media in 5 languages.

1 Introduction

Web technology has substantially evolved since its early origins in the late eighties and content management systems are becoming more and more popular among news providers, such as FINANCIAL TIMES (*http://www.ft.com/*), LE MONDE (*http://www.lemonde.fr/*), or DER SPIEGEL (*http://www.spiegel.de/*). Consequently, HTML pages tend to resemble mosaic-like patchworks made up of an entire plethora of content blocks, among which the actual textual information only represents a minor fraction. Link lists to related articles or services, advertisements, non-related images, disclaimers at the page's bottom, and readers' comments likewise populate the browser window.

However, the enrichment of pure content with page clutter may entail serious implications, in particular for search-oriented applications such as online reputation monitoring platforms [21, 6]: Those systems are geared towards recording media coverage in online news, e.g., by counting all citations of a particular brand or product. Suppose an article about Microsoft, taken from the international version of the popular business news portal FINANCIAL TIMES (*http://www.ft.com/cms/*). Next to the actual news information, there are links to many other events of the day, such as a press posting about Yahoo. When measuring the daily press coverage of the latter company, counting the depicted page as a hit is *misleading* as the actual article is *not* about Yahoo. Search engines performing full-text indexation of HTML pages cannot circumvent such issues as they do not dissect between actual content and surrounding clutter.

We propose an approach that allows for fully-automated extraction of content from HTML pages. Our system's focus is mainly on news pages, but can also be used for other Web page genres. The basic concept is to extract coherent blocks of text from HTML pages, using DOM parsing, and to compute linguistic and structural features for each block. These features are then used to decide whether the block at hand is signal or noise, based upon feature thresholding. Feature thresholds are learned using a non-linear optimization technique, known as Particle Swarm Optimization [10].

Empirical evaluations by virtue of human labelling of 610 documents taken from approximately 180 news sources in 5 Western languages have shown that our approach produces results that come close to human judgement with respect to the classification of signal and noise.

2 Related Work

Early approaches to extracting specific information from HTML Web documents date back to the late nineties and are now commonly known as wrapper induction systems

[11]. Most of these wrapper induction systems operate in a semi-automated fashion, requiring human labelling of relevant passages for each site template, e.g., product pages on Amazon.com (*http://www.amazon.com*). Hence, wrappers are learned inductively from examples. The labelling process is commonly supported by GUIs [2].

Newer breeds of wrapper generators can work in a fully-automated fashion, but only for Web pages containing numerous recurring entries of like style and format, e.g., result pages from *search engines* (see, e.g., [15] and [20]). These systems are not suitable for textual content extraction from *news* pages as news articles are made up of primarily non-recurring content blocks.

An approach to domain-oriented extraction of text content is proposed in [5]. However, human effort is required for adaptation to novel domains. Structural approaches for passage retrieval are proposed in [17] and [7]. The latter two papers make use of the organization of HTML pages, i.e., tag semantics and their nesting, in order to extract blocks of relevant content. Both do not necessitate human intervention. Gupta *et al.* [9, 8] use structural information, but proceed in a reverse fashion: instead of extracting relevant content from Web pages, their approach removes non-relevant information, so that only signal remains.

Newer proposals for content extraction take into account visual cues of Web page rendering, e.g., the distance of conceptual blocks from the screen's center, the alignment of page segments, and so forth. These systems make use of rendering engines in Web browsers, which translate directly between HTML elements and their positioning within the browsing window. Examples include [3], [16], and [19].

A number of systems also computes *features* on HTML blocks, such as [18], [12], and [13]. None of them works for extracting content from *news* Web pages, though, as they are either geared towards Web spam detection [13] or towards extraction of recurring objects [18, 12].

3 Approach Outline

Our approach to content extraction is applicable to *any* type of news pages, no adaptation or human intervention is required once feature thresholds have been learned by the Particle Swarm Optimizer. The algorithmic clockwork can be divided into three major blocks, namely the merging of text blocks on a structural level, the computation of text block features, and the decision-taking whether to keep or discard the block at hand.

The input of our system is a plain HTML page, its output given by a text document containing the extracted content.

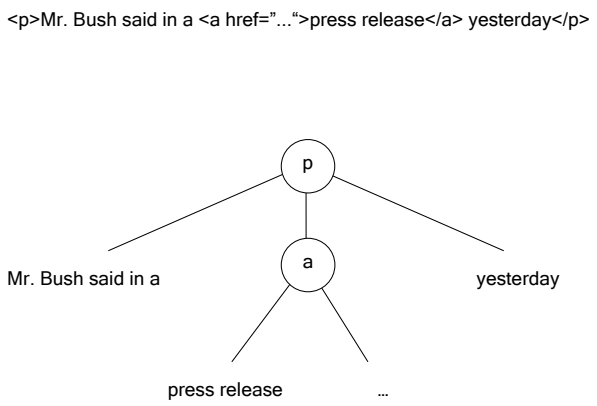

Figure 1. DOM model of an HTML fragment

3.1 Merging and Selecting Text Blocks

First, the input HTML page is transformed into well-formed XHTML. This processing step is performed by an external library and allows us to build a DOM model of the current document. Before traversing the DOM tree, spotting search blocks, several preprocessing steps take place that are aware of the semantics of HTML elements. To this end, we apply two types of operations to DOM elements and their subtrees, namely **removal** and **pruning**:[1]

Consider a fragment from an HTML page, as shown in Fig. 1: The displayed sentence is broken up into three text blocks, owing to the nested anchor tag. As we want sentences and paragraphs to be considered as *one single* block of text, we need to *remove* certain HTML elements before creating the document's DOM representation. Removal thereby means that only the element occurrence itself is discarded, but not its DOM subtree underneath.

Within the class of removed elements, another distinction is made, namely between those elements that are replaced by a whitespace character and those that are simply stripped. For instance, tag $< br >$ is replaced with a blank, for two words separated by the above element and no additional whitespace would become one after discarding $< br >$. On the other hand, font formatting tags like $< i >$, $< b >$, or $< strike >$ are simply discarded, without insertion of whitespace. Fig. 2 lists all those elements that are removed with or without whitespace insertion. While elements become removed from the document, their occurrence is nevertheless recorded, so that for any given text block, the number of its former contained elements of one given type can be determined.

Next to removing some elements, there are a number of

[1] The terms "pruning" and "removal" have been chosen so as to literally differentiate between both operations.

remove w/ space	remove w/o space	pruning
BR	A	ADDRESS
CAPTION	B	APPLET
CENTER	BIG	AREA
CITE	BLINK	BASE
DD	EM	BGSOUND
DIV	FONT	BUTTON
DL	FORM	CITE
DT	I	COL
FRAME	SMALL	COLGROUP
FRAMESET	SPAN	COMMENT
HR	STRIKE	DEL
H_	STRONG	DIR
LABEL	SUB	EMBED
LI	SUP	SAMP
NOBR	VAR	FIELDSET
OL		FRAMESET
P		HEAD
PRE		IMG
S		INPUT
SAMP		INS
U		LISTING
UL		MAP
WBR		OPTION
XMP		SCRIPT
		STYLE
		TEXTAREA

Figure 2. Categorization of HTML elements

elements, such as $< script >$ and $< img >$ that are *pruned* from the tree representation of the HTML document. That is, the respective element *and* its subtree are discarded. The rationale behind this step is that certain elements will definitely contain no actual content, such as $< script >$.

Upon removing and pruning elements, a DOM tree is eventually constructed from the processed document. Tree traversal allows us to only select the merged text nodes, as we are not interested in other information. As a result from the first step, we obtain text blocks and keep track of the number of occurrences of each type of HTML element that used to appear therein.

3.2 Computing Features

In the next step, eight features are computed for each text block. The computation hereby harnesses a range of NLP tools, such as sentence splitters and tokenizers. To this end, we used libraries from the popular GATE suite [4] and OpenNLP (*http://www.opennlp.org*). All features are numerical in essence.

3.3 Linguistic Features

Four of the overall eight features are linguistic in nature, meaning they take into account syntax and distribution of characters or tokens:

- **Average sentence length.** The block's average sentence length is supposedly higher for informative content blocks than for clutter. For instance, text within link lists is rather limited in terms of sentence length.
- **Number of sentences.** Continuous text, e.g., news articles, tends to be made up of many sentences. Hence, informative text blocks are expected to satisfy the respective threshold by exhibiting greater values.
- **Character distribution.** Operating on mere characters rather than the token level or sentence level, the feature at hand measures the ratio of alphanumeric characters within the text block as opposed to non-alphanumeric characters. Our hypothesis is that content blocks have a high ratio in favor of numbers and letters.
- **Stop-word ratio.** Stop-words are terms that occur very frequently in text and bear no intrinsic information gain. Examples in English are "this" or "she". We compute the ratio of stop-words with respect to non-stop-words using stop-word lists from 14 languages. We then take the argmax of all 14 ratios, which gives us as a by-product the language the text is written in. Our working hypothesis is that the ratio of stop-words tends to be high for continuous text, as opposed to short link lists, advertisement slogans and so forth.

3.4 Structural Features

Next to the linguistic features, we compute structural features that are based on HTML element occurrences. As mentioned earlier, these elements were recorded prior to their discarding.

- **Anchor ratio.** A high ratio of anchor-tagged words with respect to words not contained in links is clearly an indication for link lists, site navigations, etc. Hence, we expect the anchor ratio of informative content blocks below some given threshold.

- **Format tag ratio.** Formatting tags are HTML elements such as $< b >$, $< em >$, $< small >$, and so forth. We suppose continuous text to contain a considerably higher share of formatting markup with respect to the overall number of words than page clutter.

- **List element ratio.** Unordered and ordered HTML lists are typical expressions of site navigations, links to related topics, etc. On the other hand, actual textual content tends to exhibit small shares of list elements only.

- **Text structuring ratio.** Text structure markup comprises elements to indicate headlines, e.g., $< h1 >$, paragraph breaks $< p >$, and text block alignment, e.g., $< left >$. We expect that true content, such as an article, is more likely to contain structuring markup than non-informative content.

3.5 Classifier Training Framework

Having the features computed, a decision function is needed in order to determine whether to discard or keep a given text block. To this end, we have to define *thresholds* for each unique feature. As has been indicated in the preceding sections, some features are imposing minimality constraints on their respective thresholds while others demand maximality constraints. E.g., the threshold for the average sentence length is a *lower* bound. That is, when a text block's average sentence length is below the threshold, it is considered noise rather than signal.

The decision function we conceive combines features in a conjunctive fashion, i.e., only when *all* feature thresholds are satisfied, the text block will become classified as signal.

3.5.1 Benchmark Composition

In order to train the classifier and evaluate the goodness of classification at a later stage, we composed a benchmark set of 610 human-labelled documents. The set contained the original HTML documents, news articles from diverse popular media sources taken from 5 different languages, exhibiting the following make-up: 265 English documents, 195 German, 50 Spanish, 50 French, and 50 Italian Web pages. For each language, approximately 2-4 articles were taken from the same source, e.g., NY TIMES; our policy for these same-site pages was to select articles from different categories, such as sports, politics, and so forth.

The pages were hand-labelled by three computer scientists who were explained the underlying task in detail beforehand. First, the labeller would open the HTML news page and examine its appearance thoroughly. Then he would mark the text passages he thought were part of the news article and would copy them to a plain-text file. Hereby, special attention was paid to choosing the right encoding, which appeared of utter importance in particular for French and Spanish.

At this stage, it became clear that human labelling would not achieve 100% agreement when having several people label the same document. The distinction between signal and noise is non-obvious even for humans: For instance, while one labeller would include the date the article was written, another would not. So we had 2 of our labellers extract a sample of 70 documents in parallel, for the purpose of benchmarking inter-human agreement.

3.5.2 Computing Similarity

Next, a similarity function $\mathrm{sim}^{\sigma}(i,j)$ is needed in order to compare two documents d_i^{σ} and d_j^{σ}, with d_i^{σ} and d_j^{σ} representing the textual content extracted from an HTML news article page a_{σ}. Each of the two documents $d_i^{\sigma}, d_j^{\sigma}$ may likewise count as a result of human labelling as well as fully-automated content extraction. The continuous value range is in $[0,1]$, where 1 expresses identity and 0 maximal dissent:

$$\mathrm{sim}^{\sigma}(i,j) \in [0,1] \tag{1}$$

We opted for using a variant of cosine similarity, known from vector-space queries [1] in information retrieval. The eventual function looks as follows:

$$\mathrm{sim}^{\sigma}(i,j) := \delta \cdot \frac{\vec{v_i} \bullet \vec{v_j}}{\parallel \vec{v_i} \parallel \cdot \parallel \vec{v_j} \parallel}, \tag{2}$$

where

$$\delta := \min\left(\frac{\sum_k v_{i,k}}{\sum_k v_{j,k}}, \frac{\sum_k v_{j,k}}{\sum_k v_{i,k}}\right) \tag{3}$$

Vectors $\vec{v_i}$ and $\vec{v_j}$ represent document vectors for the respective documents d_i^{σ} and d_j^{σ}. For the components of these vectors, we considered all terms with length > 1, effectively removing all punctuation symbols. Factor δ is a modification to the vector-space model we implemented in order to make the cosine similarity function *scale-invariant*: The vector-space model only considers the *distribution* of term frequencies, but not their absolute values. E.g., when having two documents d_i^{σ}, d_j^{σ}, where the second one is a doubled copy of the first one so that $\forall k : v_{j,k} = 2 \cdot v_{i,k}$, the similarity function $\mathrm{sim}^{\sigma}(i,j)$ would nevertheless return 1, even though the two documents are *not* identical.

By multiplying the cosine similarity value with δ, the minimum ratio of both document lengths in terms of word tokens, we are able to circumvent this effect.

Even though our modified cosine similarity function does not consider the order of terms, we can do without such order at no expense in precision: marking and copying text during labelling inherently preserves the order of terms.

The same holds true for the extraction of content by means of our automated system, which takes text blocks in their sequential order.

Manual probing on a plurality of articles has shown that our similarity function works in an intuitive fashion and reflects the human perception of similarity of extracted content.

3.5.3 On Particle Swarm Optimization

Upon definition of a similarity function and benchmark set, learning optimal feature thresholds comes up next. The benchmark set of 610 articles and labelled documents was subdivided into a training set and test set.

The training set's purpose is to enable the determination of thresholds that allow good discrimination between true article content and page clutter. In order to determine *optimal* threshold values, an exploration of the complete state space for all eight pivot parameters would be required, which appears impossible owing to combinatoric explosion.

We therefore opted for stochastic non-linear optimization, namely Particle Swarm Optimization (PSO), a technique inspired by bird flocking and fish schooling [10]:

Sets of thresholds for all eight features are encoded as particles, and each particle, which can be seen as a vector $\vec{p}_i$ in 8-dimensional space, is part of a population of n particles. Every $\vec{p}_i$ has a velocity vector $\vec{w}_i$ and a memory of its own best state $\vec{\rho}_i$, i.e., the state for which the respective threshold vector performed best on the training set.

In each step, particles change their position in hyperspace by moving in the direction of their own best state and, at the same time, by following the current population P_k's best particle $\vec{p}_b$. The degree to which particles move towards their own best state is called the *cognitive* component, τ. The extent to which they follow the population's best particle is called the *social* component, ϕ. Typically, τ and ρ both assume values around 2.

Moreover, some randomness comes into play when computing the new velocity vector, by means of random factors r_1 and r_2. After positioning and velocity vector adaptation, a new population P_{k+1} is created.

The procedure for PSO is shown in Alg. 1, where μ denotes the fitness function that gives the particle's classification accuracy and ω defines an inertial constant usually set close to 1. Function μ works by computing similarity scores for all pairs of human-labelled and automatically extracted documents, averaged across the size of the training set.

An in-depth introduction to PSO is given in [10].

$\forall i \in \{1, 2, \ldots, n\}, j \in \{1, 2, \ldots, 8\} : p_{i,j} \leftarrow \text{rnd}_{[0,1]}$;
$\forall i \in \{1, 2, \ldots, n\} : \vec{w}_i \leftarrow \vec{0}$;

do loop
  for $i \leftarrow 1$ **to** n
    for $j \leftarrow 1$ **to** 8
      $r_1 \leftarrow \text{rnd}_{[0,1]}, r_2 \leftarrow \text{rnd}_{[0,1]}$;
      $w_{i,j} \leftarrow \omega \cdot w_{i,j} + r_1 \cdot \tau \cdot (\rho_{i,j} - p_{i,j}) + r_2 \cdot \phi \cdot (p_{b,j} - p_{i,j})$;
      $p_{i,j} \leftarrow p_{i,j} + w_{i,j}$;
    end for
    if $\mu(\vec{p}_i) > \mu(\vec{\rho}_i)$ **then** $\vec{\rho}_i \leftarrow \vec{p}_i$ **end if**;
  end for
  $\vec{p}_b \leftarrow \text{argmax}_{i \in \{1,2,\ldots,n\}}(\mu(\vec{p}_i))$;
loop until convergence

Algorithm 1: Particle Swarm Optimization

4 Training and Empirical Evaluation

In order to compare our approach's classification accuracy with an upper bound defined by inter-human consent, as well as various lower bounds, we performed empirical analyses on the test set. Before testing, the training set was used to learn near-optimal thresholds via PSO. We adopted a 70% training set versus 30% test set split on our set of 610 documents.

4.1 Training the Classifier

For the training of the classifier, we used a population size of $n = 100$ and stopped the optimization procedure upon generation of 100 populations. In order to make sure that PSO parameters, i.e., settings for n, τ, and ϕ, did not have an overly large impact on the outcome, we tried various combinations. Fig. 3 shows a comparison of two runs: For the first, the cognitive component is favored over the social component. For the second, both components are equal.

The figure shows that both PSO runs converge quickly, roughly after 10 iterations. The two performance curves for the best particle exhibit no significant differences. When cognition is favored over social behavior, bumps become more pronounced, both for the best particle curve as well as the average population performance. We believe this observation due to the increased degree of randomness introduced by having particles focus more on themselves than the best particle around.

The best particle was chosen from generation 32, from the curve where cognition outweighs social behavior. The performance of the respective particle's thresholds, averaged across all training documents, amounted to 0.827.

We also tried various other population sizes n, finding no significant differences in the results produced.

4.2 Performance Benchmarking

The classifier was parameterized with the threshold values from the best particle we found during training. For the benchmarking, we first defined the upper bound, which was set by inter-human agreement:

As has been mentioned in Sec. 3.5.1, a sample of 70 documents from the full test set was labelled by two persons in parallel, so that we could measure their mutual agreement with respect to a news page's true content versus clutter. The average score across all 70 documents was 0.909; this figure tells us that even humans do not 100% comply with respect to what is signal and what is noise. In Tab. 1, the human-only strategy is named HL.

Next, we computed the classification accuracy for our automated content extraction system (FE), based on the *full* test set. The score we obtained amounted to 0.857.

Our content extraction scheme is based on two paradigms, namely pruning and removing elements that are highly unlikely to contain informative content, as well as the feature extraction and matching procedure. In particular the pruning step substantially contributes to the removal of unwanted content, e.g., by discarding text blocks within $< script >$ elements, which are extremely unlikely to bear relevant news content. Hence, in order to assess the effect of feature extraction and matching, we tested our system with pruning and removing, but without feature matching (PR). This strategy essentially equates FE configured in such way as to accept *all* blocks that attain the feature matching step.

Strategy	HL	**FE**	PR	TB
⊘**-Score**	0.909	**0.857**	0.601	0.424

Table 1. Performance benchmark scores

The border-line is marked by strategy TB, which simply extracts all text nodes within the original HTML document, operating on the DOM tree model.

Scores for PR and TB were considerably lower than for FE (see Tab. 1). On the other hand, FE comes close to the human upper bound. This result shows that our automated system for content extraction, using feature extraction and feature matching against learned thresholds, exhibits a very good classification behavior.

4.3 Feature Entropy Evaluation

Besides assessing our novel approach through benchmarks, we were also interested in understanding the utility and impact of the various features we used.

To this end, we performed evaluations on the complete test set, testing each feature in isolation, i.e., without computing and comparing the other seven features against their respective thresholds. Hereby, we discretized the continuous value range of each feature in a suitable fashion and used each of these discrete data points as threshold value for classification. We then recorded the performance of each single-feature classifier and each threshold and plotted the results into charts (see Fig. 4).

Top classifier accuracy amounted to 0.77, attained by means of classification by the maximum allowed ratio of anchor tags, as shown in Fig. 4(b). Using the minimum ratio of stop-words as decision function appeared to be the second-best classifier, followed by the feature for minimum number of sentences. On the other hand, the list tag ratio, formatting tag ratio, and text structuring ratio exhibited poor performance, not too far away from the PR strategy's accuracy.

Hence, apart from the anchor tag ratio, we found that linguistic features exhibit lower, i.e., better, entropy than structural ones. Moreover, as the single-classifier experiment has shown, the combination of features boosts classifier performance dramatically, providing lift from 0.77 to 0.857.

4.4 Outlook and Conclusion

Content extraction from news articles on the Web is a non-trivial, though indispensable task for many applications. Page clutter abounds and an automated extraction scheme needs to be implemented in order to handle the vast number of news sources to be monitored.

We have presented a novel paradigm that works in a fully-automated fashion, classifying text blocks within HTML pages as signal or noise by means of linguistic and structural features. Performance evaluations have shown that our approach exhibits an accuracy that comes close to human judgement.

We plan to extend our approach in diverse directions. First, we would like to include more features. For instance, we conjecture that part-of-speech information could provide a fruitful basis for several new features leveraging high information gain.

Moreover, besides techniques borrowed from non-linear stochastic optimization, we intend to use techniques from machine learning, e.g., C4.5 decision trees [14], as means for defining thresholds and weighing their precedence within the decision-making process.

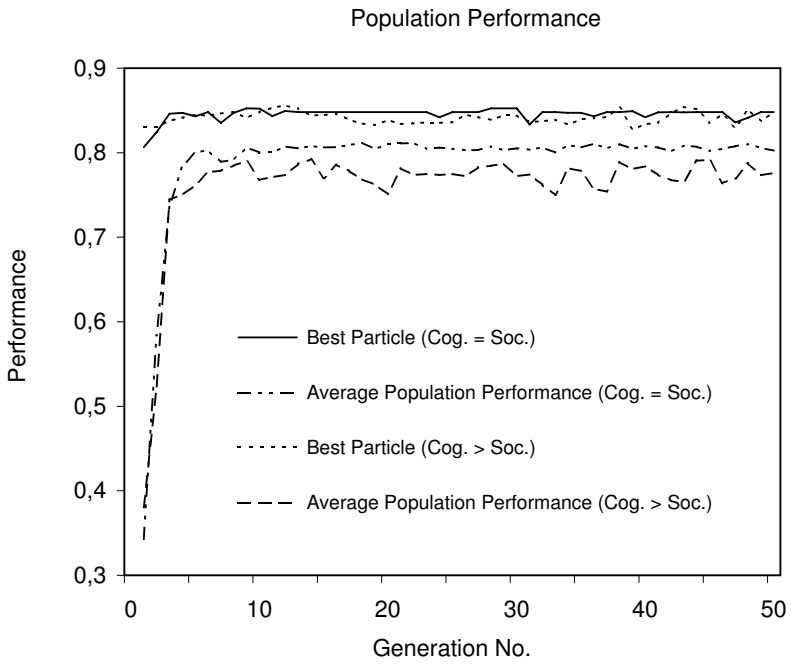

Figure 3. PSO training results

From a more paractical point of view, the content extraction system presented in this paper will be incorporated as processing component into our reputation monitoring platform used by Siemens Corporate Communications.

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, Reading, MA, USA, May 1999.

[2] R. Baumgartner, S. Flesca, and G. Gottlob. Visual Web information extraction with Lixto. In *Proceedings of the 27th International Conference on Very Large Databases*, pages 119–128, Roma, Italy, 2001.

[3] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Block-based Web search. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 456–463, Sheffield, UK, 2004. ACM Press.

[4] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: An architecture for development of robust HLT applications. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 168–175, Philadelphia, PA, USA, 2001. Association for Computational Linguistics.

[5] D. de Castro Reis, P. Golgher, A. Silva, and A. Laender. Automatic Web news extraction using tree edit distance. In *Proceedings of the 13th International Conference on World Wide Web*, pages 502–511, New York, NY, USA, 2004. ACM Press.

[6] N. Glance, M. Hurst, and T. Tomokiyo. Blogpulse: Automated trend discovery for weblogs. In *Proceedings of the WWW 2004 Workshop on the Weblogging Ecosystem*, New York, NY, USA, 2004.

[7] J. Goller. STAN: Structural analysis for Web documents. In *Proceedings of the Second Internation Workshop on Web Document Analysis*, pages 15–18, Ediburg, UK, 2003.

[8] S. Gupta, G. Kaiser, P. Grimm, M. Chiang, and J. Starren. Automating content extraction of HTML documents. *World Wide Web*, 8(2):179–224, 2005.

[9] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. DOM-based content extraction of HTML documents. In *Proceedings of the 12th International Conference on World Wide Web*, pages 207–214, Budapest, Hungary, 2003. ACM Press.

[10] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ, USA, 1995. IEEE Computer Society.

[11] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *International Joint Conference on Artificial Intelligence*, pages 729–737, Nagoya, Japan, 1997. Morgan Kaufmann.

[12] S.-H. Lin and J.-M. Ho. Discovering informative content blocks from Web documents. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 588–593, Edmonton, Alberta, Canada, 2002. ACM Press.

[13] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam Web pages through content analysis. In *Proceedings of the 15th International Conference on World Wide Web*, pages 83–92, Edinburgh, Scotland, 2006. ACM Press.

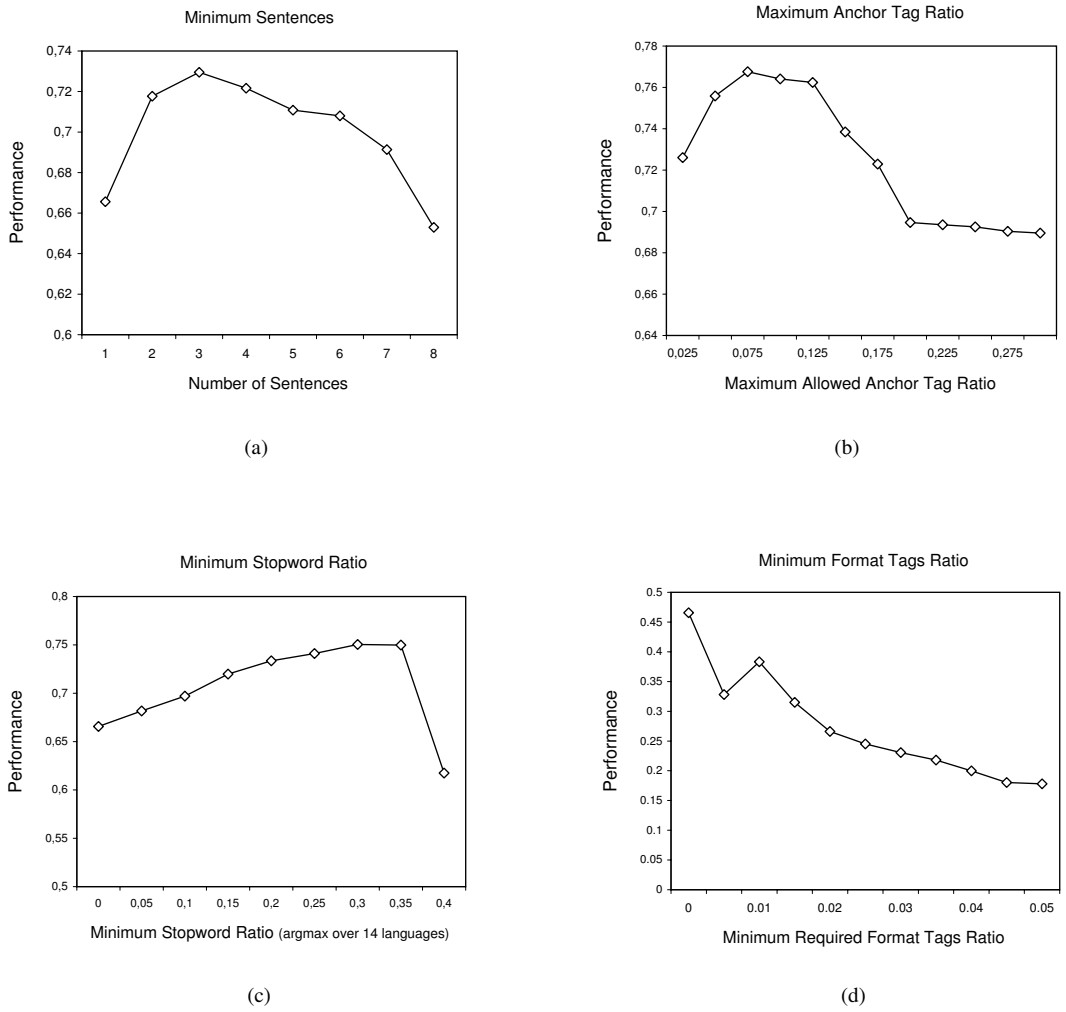

Figure 4. Single-feature classifier performance scores

[14] R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[15] K. Simon and G. Lausen. VIPER: Augmenting automatic information extraction with visual perceptions. In *Proceedings of the 2005 ACM CIKM Conference on Information and Knowledge Management*, pages 381–388, Bremen, Germany, November 2005. ACM Press.

[16] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma. Learning block importance models for Web pages. In *Proceedings of the 13th International Conference on World Wide Web*, pages 203–211, New York, NY, USA, 2004. ACM Press.

[17] A. Sun and E.-P. Lim. Web unit mining: Finding and classifying subgraphs of Web pages. In *Proceedings of the 12th International Conference on Information and Knowledge Management*, pages 108–115, New Orleans, LA, USA, 2003. ACM Press.

[18] Y.-F. Tseng and H.-Y. Kao. The mining and extraction of primary informative blocks and data objects from systematic Web pages. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 370–373, Hong Kong, China, 2006. IEEE Computer Society.

[19] Y. Yang and H.-J. Zhang. HTML page analysis based on visual cues. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 859–864, Washington, DC, USA, 2001. IEEE Computer Society.

[20] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *Proceedings of the 14th International Conference on World Wide Web*, pages 76–85, Chiba, Japan, 2005. ACM Press.

[21] C.-N. Ziegler and M. Skubacz. Towards automated reputation and brand monitoring on the web. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 1066–1070, Hong Kong, China, December 2006. IEEE Computer Society Press.