Web Page Cleaning with Conditional Random Fields

Michal Marek [1], Pavel Pecina [1], Miroslav Spousta [1]

Abstract

This paper describes the participation of the Charles University in Cleaneval 2007, the shared task and competitive evaluation of automatic systems for cleaning arbitrary web pages with the goal of preparing web data for use as a corpus in the area of computational linguistics and natural language processing. We try to solve this task as a sequence-labeling problem and our experimental system is based on Conditional Random Fields exploiting a set of features extracted from textual content and HTML structure of analyzed web pages for each block of text. Labels assigned to these blocks then discriminate between different types of *content blocks* containing useful material that should be preserved in the document and *noisy blocks* of no linguistics interest that should be eliminated.

Keywords : Web page cleaning, Sequence labeling, Conditional random fields.

1. Introduction

The web with its enormous amounts of textual materials is a rich and easily accessible source of linguistic data that can be used to create extremely large corpora with relatively low cost and within a short period of time (compared to the traditional way of building text corpora which is an expensive and time-consuming process). The idea of having a corpus "as large as the web" recently attracted attention of many researchers in computational linguistics, natural language processing, and related areas, who would greatly benefit from such amount of data. Creating such a corpus comprises two steps: a) *web crawling* - automatic browsing the web and keeping a copy of visited pages and b) *cleaning* the pages to be included in the corpus. In this work we focus on the latter.

Apart from the main content, a typical web page contains other material of no linguistic interest, such as navigation bars, panels, and frames, page headers and footers, copyright and privacy notices, advertisements and other data (often called *boilerplate*). The goal is to detect and remove such parts from an arbitrary web page.

Although this task might seem too heterogeneous for methods based on supervised training to be appropriate, we solve it as a sequence-labeling problem with Conditional Random Fields trained on a manually cleaned set of web pages. Our primary goal is an attempt to explore whether supervised training methods can perform well enough to be successfully used in this task.

[1] Institute of Formal and Applied Linguistics, Charles University, Prague, Czech Republic, {marek,pecina,spousta}@ufal.mff.cuni.cz

2. Related Work

Although web page cleaning is a crucial step in the procedure of building a web corpus, only a relatively little work has been done in this area. Most of it originated in the area of web mining and search engines, e.g. (Cooley *et al.* 1999) or (Lee *et al.* 2000). In (Bar-Yossef & Rajagopalan 2002), a notion of pagelet determined by the number of hyperlinks in the HTML element is employed to segment a web page; pagelets whose frequency of hyperlinks exceeds a threshold are removed. (Lin & Ho 2002) extract keywords from each block content to compute its entropy, and blocks with small entropy are identified and removed. In (Yi *et al.* 2003) and (Yi & Liu 2003), a tree structure is introduced to capture the common presentation style of web pages and entropy of its elements is computed to determine which element should be removed. In (Chen *et al.* 2006), a two-stage web page cleaning method is proposed. First, web pages are segmented into blocks and blocks are clustered according to their style features. Second, the blocks with similar layout style and content are identified and deleted. As far as the authors know, none of the published method was based on sequence labeling or a similar learning algorithm.

3. System Overview

The system for web page cleaning presented in this paper is based on sequence labeling with CRF++ [1] implementation of Conditional Random Fields (Lafferty *et al.* 2001). It is aimed at cleaning arbitrary web pages as specified in the Cleaneval shared task description [2]. The cleaning process consists of several steps:

1) Standardizing HTML code

The raw HTML input is passed through Tidy [3] in order to get a valid and parsable HTML tree. During development, we found only one significant problem with Tidy, namely interpreting JavaScript inside the `<script>` element, and employed a simple workaround for it in our system. Except for this particular problem which occurred only once in our training data, Tidy has proved to be a good choice.

2) Precleaning

Afterwards, the HTML code is parsed and parts that are guaranteed not to carry any useful text (e.g. scripts, style definitions, embedded objects, etc.) are removed from the HTML structure. The result is a valid HTML code.

3) Text block identification

In this step, the precleaned HTML text is again parsed with a HTML parser and interpreted as a sequence of text *blocks* separated by one or more HTML tags. For example, the snippet ‘‘`<p>Hello <b>world</b>!</p>`’’ would be split into three blocks, ‘‘`Hello`’’, ‘‘`world`’’, and ‘‘`!`’’. Each of the blocks is then a subject of the labeling task and cleaning.

[1] http://crfpp.sourceforge.net/
[2] http://cleaneval.sigwac.org.uk/
[3] http://tidy.sourceforge.net/

4) Feature extraction

In this step, a feature vector is generated for each block. The list of features and their detailed description is presented in the next section. The features must have a finite set of values [4]. The mapping of integers and real numbers into finite sets was chosen empirically and is specified in the configuration. Most features are generated separately by independent modules. This allows for adding other features and switching between them for different tasks.

5) Learning

In accordance to the Cleaneval annotation guidelines and definition of the markup tags, each *content block* occurring in our training data was manually assigned one of the following labels: *header*, *paragraph*, *list*, and *continuation* for blocks that are in fact continuations of preceding content blocks of any type. Other blocks are considered as *noisy blocks* and are assigned the label *other*.

The sequence of feature vectors including labels extracted for all blocks from the training data are then transformed into the actual features used for training the CRF model according to offset specification described in a *template* file.

6) Cleaning

Having estimated parameters of the CRF model, an arbitrary HTML file can be passed through steps 1–4, and its blocks can be labeled with the same set of labels as described above. These automatically assigned labels are then used to produce a cleaned output. Blocks labeled as *header*, *paragraph*, or *list* remain in the document and are marked by a corresponding markup tag (`<h>`, `<p>`, `<l>`), blocks labeled as *continuation* remains in the document as they are (no tag is inserted) and blocks labeled as *other* are deleted.

4. Features

Features recognized by the system can be divided by their scope into three subsets: features based on the HTML markup, features based on textual content of the blocks, and features related to the document.

Markup-based Features

container.p, container.a, container.u, container.img, container.class-header, container.class-bold, container.class-italic, container.class-list, container.class-form

For each parent element of a block, a corresponding *container.** feature will be set to 1, e.g. a hyperlink inside a paragraph will have the features *container.p* and *container.a* set to 1. This feature is especially useful for classifying blocks: For instance a block contained in one of the `<hx>` elements is likely to be a header, a *container.li* feature suggests a list item, etc. The *container.class-** features refer to classes of similar elements rather than to elements themselves.

[4] This is a limitation of the CRF tool used.

split.p, split.br, split.hr, split.class-inline, split.class-block

For each opening or closing tag encountered since the last block, we generate a corresponding *split.** feature. This is needed to decide, whether a given block connects to the text of the previous block (classified as *continuation*) or not. Also, the number of encountered tags of the same kind is recorded in the feature. This is mainly because of the `<br>` tag; a single line break does not usually split a paragraph, while two or more `<br>` tags usually do. The *split.class-** features again refer to classes of similar elements.

Content-based Features

char.alpha-rel, char.num-rel, char.punct-rel, char.white-rel, char.other-rel

These features represent the absolute and relative counts of characters of different classes (letters, digits, punctuation, whitespace and other) in the block.

token.alpha-rel, token.num-rel, token.mix-rel, token.other-rel token.alpha-abs, token.num-abs, token.mix-abs, token.other-abs

These features reflect counts distribution of individual classes of tokens[5]. The classes are words, numbers, mixture of letters and digits, and other.

sentence.count

Number of sentences in a block. We use a naive algorithm basically counting periods, exclamation marks and question marks, without trying to detect abbreviations. Given that the actual count is mapped into a small set of values anyway, this does not seem to be a problem.

sentence.avg-length

Average length of a sentence, in words.

sentence-begin, sentence-end

These identify text blocks that start or end a sentence. This helps recognizing headers and list items (as these usually do not end with a period) as well as continuation blocks (*sentence-end=0* in the previous blocks and *sentence-start=0* in the current block suggest a continuation).

first-duplicate, duplicate-count

The *duplicate-count* feature counts the number of blocks with the same content (ignoring white space and non-letters). The first block of a group of twins is then marked with *first-duplicate*. This feature serves two purposes: On pages where valid text interleaves with noise (blogs, news frontpages, etc), the noise often consists of some phrases like “read more...”, “comments”, “permalink”, etc, that repeat multiple times on the page. The second purpose of this feature is to identify

[5] Tokens being sequences of characters separated by whitespace for this purpose.

quotes in discussions, which are deleted in the Cleaneval development set. The *first-duplicate* feature is then used to recognize the original post.

regexp.url, regexp.date, regexp.time

While we try to develop a tool that works independently of the human language of the text, some language-specific features are needed nevertheless. The configuration defines each *regexp.** feature as an array of regular expressions. The value of the feature is the number of the first matching expression (or zero for no match). We use three sets of regular expressions: to identify times and dates (lines with creation date/time are marked as *header* in Cleaneval data) and URLs (these are usually cleaned away in the Cleaneval data).

bullet

This is a specialized version of the *regexp.** features, matching different patterns that suggest a list item (number or a single letter, closing parenthesis, dash, etc). Each combination of matches results in a different value of the feature, so that lists can be recognized as a sequence of blocks with the same *bullet* value.

div-group.word-ratio, td-group.word-ratio

The layout of many web pages follows a similar pattern: The main content is enclosed in one big `<div>` or `<td>` element, as are the menu bars, advertisements etc. To recognize this feature and express it as a number, the parser groups blocks that are direct descendants of the same `<div>` element (`<td>` element respectively). A direct descendant in this context means that there is no other `<div>` element (`<td>` element respectively) in the tree hierarchy between the parent and the descendant. For example in this markup

```
<div> a <div> b c </div> d <div> e f </div> g </div>
```

the *div-group*s would be (a, d, g), (b,c) and (e, f). The *div-group.word-ratio* and *td-group.word-ratio* express the relative size of the group in number of words. To better distinguish between groups with noise (e.g. menus) and groups with text, only words not enclosed in `<a>` tags are considered.

Document-related features

position

This feature reflects a relative position of the block in the document (counted in blocks, not bytes). The rationale behind this feature is that parts close to the beginning and the end of documents usually contain noise.

document.word-count, document.sentence-count, document.block-count

This feature represents the number of words, sentences and text blocks in the document.

document.max-div-group, document.max-td-group

The maximum over all *div-group.word-ratio* and a maximum over all *td-group.word-ratio* features. This allows us to express "fragmentation" of the document – documents with a low value of one of these features are composed of small chunks of text (e.g. web bulletin boards).

5. Data and Evaluation

For our development purposes we had 104 manually cleaned HTML documents available: 51 of them were provided by Cleaneval and the rest was randomly selected, downloaded, and cleaned following the same guidelines by a volunteer. In this development data set, 22 501 blocks were identified and assigned appropriate labels. Their distribution is shown in the following table.

label	count
header	1 996
list	1 149
paragraph	3 419
continuation	3 380
total (*content*)	9 944
other (*noise*)	12 557

Table 1. Labels distribution in the development data set.

The data set was randomly split into six subsets and for better estimation of performance measures used in a six-fold crossvalidation. Evaluation measures were of two types:
a) labeling accuracy – the ratio of correctly assigned block labels (1) from the full label set and (2) distinguishing only between *content* and *noisy* blocks;
b) cleaning performance – the official Cleaneval scoring measures based on (3) edit distance and the extent to which the markup tags indicate blocks of text starting and ending in the same place and (4) alignment of text alone, ignoring the markup tags plus (5) a combination of the latter two referred as the total score.

6. Experiments and Results

First, we have performed a series of experiments where we disabled each of the 46 features one by one with the following observations:

The variance among the results on the six crossvalidation subsets was relatively high in all experiments; the total score ranged from 66% to 80%. This is partially caused by the relatively small amount of training and test data in each run but it also proves the heterogenous character of the task.

In contrast, the difference among the results of all experiments was relatively low. The total scores ranged from 71.9% to 74.5%. A possible explanation is a certain redundancy in the feature set.

Disabling some of the features slightly improved the results. The total score with all features enabled was 73.9%, while disabling the *document.word-count* feature resulted in a score of 74.6%. Although this difference is probably not statistically significant we used this criterion and disabled some features for cleaning the evaluation data.

We also performed two other experiments: one with only the content-based features enabled and one with only the markup-based features enabled. Surprisingly, the results only dropped to 65.3% for markup-only and 66.5% for content-only features. This gives us an idea how much information is taken from these two types of features and how much we gain from their combination.

For the Cleaneval evaluation, we have chosen three experimental settings: Exp-1 with all features enabled, Exp-2 with the features *word-ratio*, *numeric-count*, *mixed-count* and *nonword-count* disabled, and Exp-3 with two more additionally disabled features *regexp.url* and *document.word-count*. The cross-validated results on the development data set for these experiments are displayed in Table 2.

	labeling accuracy (%)		cleaning performance (%)		
	(1) full set	(2) content-noise	(3) markup-only	(4) text-only	(5) total
Exp-1	74.45	82.60	66.71	81.11	73.92
Exp-2	75.09	83.09	67.31	81.68	74.50
Exp-3	75.01	82.88	68.46	81.83	75.15

Table 2. Labeling accuracy and cleaning performance on the development data.

7. Conclusion

We proposed a method for web page cleaning based on sequence labeling with Conditional Random Fields and presented a few initial experiments evaluated on the development data for Cleaneval 2007. With very limited costs and manual work we were able to achieve encouraging results. Using supervised training methods seems as a reasonable approach and we believe that a better set of features can bring additional performance improvement.

Acknowledgments

This work has been supported by the Ministry of Education of the Czech Republic, projects MSM 0021620838 and the Czech Science Foundation, project 201/05/H014.

References

BAR-YOSSEF Z. et RAJAGOPALAN S. (2002), "Template detection via data mining and its applications", in *WWW '02: Proceedings of the 11th international conference on World Wide Web*.

CHEN L., YE S. et LI X. (2006), "Template detection for large scale search engines", in *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*,Dijon, France.

COOLEY R., MOBASHER B. et SRIVASTAVA J. (1999), "Data Preparation for Mining World Wide Web Browsing Patterns", in *Knowledge and Information Systems*, n° 1, vol. 1.

LAFFERTY J., MCCALLUM A. et PEREIRA F. (2001), "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data", in *Proc. 18th International Conf. on Machine Learning* : Morgan Kaufmann, San Francisco, CA, USA.

LEE M.-L., LING T. W. et LOW W. L. (2000), "IntelliClean: a knowledge-based intelligent data cleaner", in *Knowledge Discovery and Data Mining*.

LIN S.-H. et HO J.-M. (2002), "Discovering Informative Content Blocks from Web Documents", in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*,Edmonton, Alberta, Canada.

YI L. et LIU B. (2003), "Web Page Cleaning for Web Mining through Feature Weighting", in *Proceedings of Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*,Acapulco, Mexico.

YI L., LIU B. et LI X. (2003), "Eliminating Noisy Information in Web Pages for Data Mining", in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*,Washington, DC, USA.