[DSB+08] *Renato Dominguez Garcia, Philipp Scholl, Doreen Böhnstedt, Christoph Rensing, Ralf Steinmetz;* **Towards To an Automatic Web Genre Classification.** no. TR-2008-10, November 2008.

Technische Universität Darmstadt

Department of Electrical Engineering and Information Technology
Department of Computer Science (Adjunct Professor)
Multimedia Communications Lab
Prof. Dr.-Ing. Ralf Steinmetz

Towards To an Automatic Web Genre Classification

Technical Report

by

Renato Domínguez García, Philipp Scholl, Doreen Böhnstedt, Christoph Rensing, Ralf Steinmetz

First published: 15 December 2008
Last revision: 15 December 2008
For the most recent version of this report see
ftp://ftp.kom.tu-darmstadt.de/pub/TR/KOM-TR-2008-10.pdf

KOM-TR-2008-10

Table of Content

1 Introduction

The number of web pages on the World Wide Web (WWW) is increasing daily. To find web pages that contain information satisfying our requirements (like taking about a specific subject or having research quality standards) is getting more and more difficult, because nearly all retrieval processes are topic-centered: We type a keyword in a web search engine in order to get the desired information but we do not only obtain relevant documents but also a lot of irrelevant documents. Therefore, there is an urgent need for methods to increase the quality of information retrieval: For example, forums are often used to find answers to typical problems like computer problems. If you have such a problem, it could be easier and faster to search only in forums. We see also that users are sometimes interested in certain *kinds* of information, or as it are called here, in particular genres [SM06].

Web genres describe and classify web documents by their characteristics like style, readability, structure, layout and meta-content. Genres of web pages can be also understood as differentiation between blogs, forums, private homepages or research web pages, for example. Topic and Genre are orthogonal. For example, the web page of a newspaper contains information about many topics ("Olympic games", "Dalai Lama", "Madonna"), but it belongs to one or more web genres ("newspaper", "blogs") [S07].

The World Wide Web contains a lot of different genres; these genres are very heterogeneous, not only in their layout but in their styles, code structure, embedded code, embedded multimedia content, use, functionality, etc. There are various classifications of web genres in related work. In [SM04], Stein et al. differentiate between eight web genres (help pages, article pages, discussion pages, shops pages, non-private portrayals, private portrayals, link collections and download pages). Dewdney et al. [DVM01] focus on the web genres advertising, bulletin board, FAQ, message board, radio news, television news and Reuter's newswire. Santini's [S04, S06, S07] classification consist of seven web genres (blog, e-shop, FAQs, online front-page, listings, personal homepage and search engine pages) after having done a study on user's expectation. In our work we focus on pages with user-generated content like blogs, wiki pages and forums. These genres are especially in expansion nowadays, because they are part of the trend in the new use of the World Wide Web technology (Web 2.0). Each web genre has a specific function: In a blog, a user, i.e. the blog's owner, can write about one or more specific topics and other users can comment on the owner's article, wiki pages allow many different users to write and modify a document collaboratively and forum pages can be used by many users to discuss about various topics.

Classification of documents has been considered many times over the past years and can be carried out according different criteria like topic, author or genre [S04a]. The goal of document classification is to determine the category of a document using the attributes of the document (called features). Manual classification is slow and costly since it relies on manpower and is simply not feasible on the web.

Hence, the tendency is now to use automatic classification using machine learning algorithms with a supervised approach [S04b]. To detect a genre using machine learning methods, a web-page is represented by the values of features that express certain attributes of genres, thus it is assumed that documents in the same genre have similar values. A classifier learns the characteristics of the different genres from a set of pre-classified examples and guesses the genre of a document based upon these values. Different machine learning techniques are explained in detail in [M97].

Some previous studies ([SM04], [S04a], [DVM01], [NKTM00]) have suggested useful features for identifying a genre of web documents, but most of them only extend the features for text documents ([KNS97], [FK06], [F48]). Unlike these approaches we focused on developing features that apply primarily to web documents.

With the expansion of the World Wide Web, web pages get progressively more structured, because of the huge number of web-based editors, which content creators use for publishing information on templated web pages. These editors are mostly used in Web 2.0 applications. Therefore, we decided to use the structure of web pages to determine the genre of a page. To extract the structure of a web page we consider its HTML code and how this code is organized. Structural features are based also on the structure of the source code of a page unlike the usual features in the literature. Many other authors tried to classify web genres using another kind of structural features like number of hyperlinks to the same/different domain [VLG07] or number of HTML tags [S06a]. We propose to use structural features that use *structural patterns* in order to classify web pages. Structural patterns are mark-up fragments of a web page that are re-occurring in the page's HTML-source twice or more often. For example, comments in a blog or in a forum are patterns, because they have a similar HTML code. With this approach it is possible to avoid using linguistic features like Part-Of-Speech-Features (POS), as these are computationally costly to extract and result in many different features [FB06]. Linguistic features are based on the singularity of the language. Avoiding linguistic features allows genre detection independently of the language of the web pages, additionally. Another advantage of this approach is that the number of features will be reduced considerably. The importance of automatically classification of web-pages is growing, because it can be helpful in order to improve the quality of information retrieval. Furthermore classification of web pages can be applied in other fields like Community Mining.

An approach based on structural features has some disadvantages too. Old web pages do not have a clear structure, because it was not common a few years ago to use templates in order to generate web pages. Web pages have been created mostly using offline editors like Notepad for Windows. Only few patterns can be found in these web pages and their hierarchical structure of HTML-files is very flat. A flat structure reduces the possibility to find patterns.

In this Technical Report we present an overview over structural features, describing the method to detect and extract patterns in Section 2. In particular, we present our definition for the term "similar

patterns" and explain the functionality of our pattern-extraction algorithm. In Section 3 we present our structural features in more detail. We describe the development process of the features and using examples showing cases in which these features can help to determine the genre of a web page. Section 4 summarizes some of the results that we achieved using our new structural features. Finally, in Section 5 we give a conclusion of this report.

Pattern-Extraction Algorithm

In this section we explain how we extract re-occurring hierarchical HTML element structures – called structural patterns – from the HTML's mark-up. HTML-files do not obey a strict structure like a predetermined organization; hence we speak about *semistructured data.* It is very usual for semistructured data to share some common structures. Consider, for example, that the web pages of faculty members in a university are constructed in the same way (e.g. biographical information, projects, publications, etc). These pages are always similar but not identical. As we do not need exact matches by the detection of frequent structures, we introduce fuzziness to the pattern recognition that allows matching of fragments even if some child tags are interchanged or the number of the tags is not the same. For example: If we take a blog post page with its comments, we will recognize the comments as patterns, because they share a similar structure. Even if the comments are different in some aspects like fonts, providing links or number of paragraphs, these comments should be recognized as patterns. Patterns exist only in the same hierarchical level of the Document Object Model[1] (DOM) tree. DOM is an object model for representing HTML or XML documents. It represents the structure of a document as a tree[2]. Further, two patterns are in the same hierarchical level, if they have the same parent node.

In order to calculate the similarity between patterns we use a method described in [RMS06], a simple approach for effective detection of the structural similarity between XML documents. This approach is based on the intuition that two HTML element structures are similar if a large fraction of their paths in the DOM-tree are the same or similar.

A HTML document can be modelled as a graph [PGW95] or a tree [WL00], where each node represents a HTML tag and each edge a hierarchical relationship between two elements. In this work, we refer to this tree description as a *structure tree.* For example, Figure 1 shows the structure tree for the following HTML document:

```
<html>
        <head>
                <title> Title of page </title>
        </head>
        <body>
                This is my first homepage.
                <b> This text is bold </b>
        </body>
</html>
```

Listings 1: Example of a HTML page

---

[1] http://www.w3.org/DOM/ [online 10.06.2008]

[2] http://en.wikipedia.org/wiki/Tree_%28graph_theory%29 [online 05.06.2008]

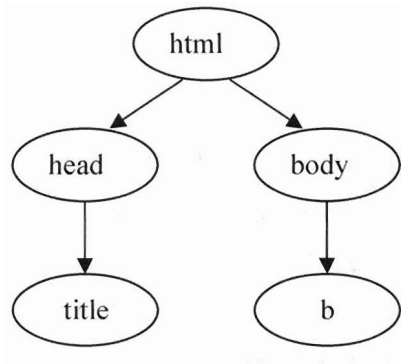

Figure 1: A structure tree

Our pattern-extraction algorithm recursively visits all nodes in a structure tree and tries to detect re-occurring fragments of HTML code (patterns). There are other approaches to calculate the similarity between trees ([GZZCK05], [KNM05], [JWZ95]), but the problem of finding the edit distance [KLE98] between unordered trees is NP-complete [ZSS92]. The chosen approach proposes a method to avoid this tree matching problem by representing a document as a set of paths. First they extract the *root paths* from the structure tree of a document. Root paths are all paths that start from the root and end at a leaf. For example, the root paths of our HTML example are "*html/head/title*" and "*html/body/b*".

We calculate the root paths for two different *pattern candidates*. Pattern candidates are nodes in the structure tree; we recursively take two nodes in the same hierarchical level as candidates and check how similar are these candidates, i.e. not only for the first two children of the HTML tag (<html>), but also for each pair of children of the same parent node. But the approach does not only compare the *root paths* of two different pattern candidates: In order to find similar paths too, all subpaths of the root paths will be extracted. Subpaths are paths in a bigger path that do not begin in the root node. The union between the root paths and their subpaths is called the *path set* of a document. Thus, the path set of our working example may be represented: {*html/head/title, html/body/b, html/head, head/title, html/body, body/b, html, head, title, body, b*}.

Given two pattern candidates, i.e. nodes and their children nodes, we want to find out if the two candidates are structurally similar. So we build the path sets and compare them with standard vector comparison techniques like Cosine measure, Dice's coefficient or Jaccard Index. Two nodes (and their children) are similar, if the cosine measure between both paths sets is bigger than a chosen threshold value. In our approach we choose 0.99 as our threshold value, i.e. two paths sets have to be at least 99% similarly.

We observed that some pages are structured as tables and some of these tables have tables inside them. In this kind of pages, we found out that many nodes are similar, just because of the table-in-table structure. So we decided to do some experiments modifying the path set. The first experiment consisted in only the paths that begin on the root node together with the nodes alone. For example, in

our working example we have { *html/head/title, html/body/b, html/head, html/body, html, head, title, body, b*}. The second experiment consisted in only considering the paths that begin on the root-node, e.g. { *html/head/title, html/body/b, html/head, html/body*} in our working example.

By these experiments we found out, that if we apply stricter rules to the path set, then we get smaller similarity values. These smaller similarities do not lead necessarily to a better accuracy (compared to our results in chapter 4): In order to get better results, we have to choose a smaller threshold value; doing this, we get similar results like with using the standard approach with our threshold value. Our experimented showed that a modification of the path set do not lead to better results, but we cannot exclude the possibility that other means of representing subtrees and measuring similarity provide better results.

HTML files consist of HTML-tags and content. Building root paths or path sets of web pages is extremely costly, because of the immense number of paths. In order to reduce this complexity we introduce some heuristics. These heuristics contribute to a significant decrease of calculation time. Usually we would have to consider all tags as pattern candidates, but there are six cases that we ignore:

1. *Comment-Tags:* Comments are tags that do not belong to the web page. They are just there as help for programmers.
2. *Processing Instruction tags*[3]*:* Tags that belong to this class are just information for the application. Like comments, they are not textually part of the web page.
3. *Inline tags:* Inline tags are used to format the content and do not play a crucial role in the structure of a document. Examples for inline tags are: `<code>`, `<dfn>`, `<em>`, `<kbd>`, `<strong>`, `<samp>`, `<var>`, `<a>` and `<bdo>`.
4. *Pseudo-block tags:* Pseudo-block tags can not be used as pattern candidates, because they are often only part of blocks. For example, the `<td>`-tag represent just a line inside a table, thus we will have too many false similarities, if we would consider it as the beginning of a pattern candidate. On the other side pseudo-block tags can be found in several pages and do no say anything about the structure of a web page. `<dd>`, `<dt>`, `<frameset>`, `<li>`, `<tbody>`, `<td>`, `<tfoot>`, `<th>`, `<thead>` and `<ul>` are examples for pseudo-block tags.
5. *Tags without children-tags:* Two tags without children but with the same name are always equal. Hence, these tags do not give additional information in order to detect patterns in web pages. Examples for tags without children-tags are `<title>` and `<b>` in Fig. 1.

---

[3] http://www.w3.org/TR/REC-xml/#sec-pi [online 04.07.2008]

6. *<p>-tags:* This tag indicate only the start of a new paragraph. <p>-tags are very common, thus they are not helpful in web page categorization.

3 Structural Features

In the last section, we showed how our pattern algorithm finds patterns in a HTML document. In this section we want to give a detailed overview over our newly proposed structural features. We developed a total of nine novel structural features. Each feature will be explained in its own subsection using the following three examples: The TechCrunch Blog[4], the Forum of the German Journal Spiegel[5] and the Fort Thunder Wikipedia page[6].

3.1 Number of detected patterns

We count the number of detected patterns in a document. The reason behind this feature is that the HTML structure of blogs and forums is very particular: Usually, we will have a list of comments as part of the blog/forum. Each of the comments in this list was created using an input mask of the blog/forum provider. Hence these comments will have a large set of similar HTML tags. In contrast to that, wiki pages are almost pattern-free as they normally represent a single article. They only follow some wiki guidelines depending on the chosen system.

Figure 2: Screenshot of a blog

[4] http://www.techcrunch.com/ [online 28.08.2008]

[5] http://forum.spiegel.de/ [online 28.08.2008]

[6] http://en.wikipedia.org/wiki/Fort_Thunder [online 28.08.2008]

How this feature can help in classification of web pages is shown in the following figures. Figures 2, 3 and 4 show screenshots of our examples and you can see immediately that the first two pages follow a strict structure (using tables), but the wiki page does not have a clear structure. These features make it easy for the pattern algorithm to find many patterns by the first two pages and only few in the third page.

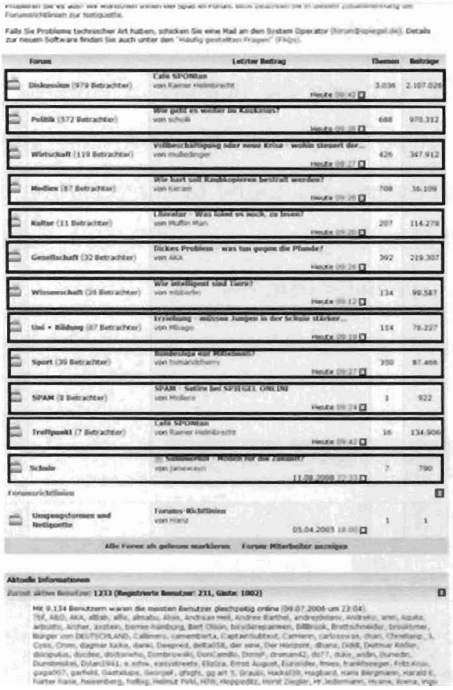

Figure 3: Screenshot of a forum

Figure 4: Screenshot a wiki page

We can also see, that the number of patterns in a HTML documents can be helpful in order to differentiate between blog and forum on one side and wiki pages on the other side. However, based on this feature only, it can not be distinguished definitely between blogs and forum. So it has to be complemented with other features.

3.2 Number of outer patterns

Some web pages have patterns inside other patterns, i.e. these patterns are in a lower hierarchy level in the DOM tree: For example, many forum pages allow the user to quote many different comments of other users. In this case, the quoted comments inside the comments have a similar structure. If we consider the comments of different users and the quoted comments as patterns, then we see that we have patterns in patterns.

To deal with this kind of documents we distinguish between the number of patterns in the highest DOM tree level and all other patterns. This feature helps us differentiating between blogs and forums. We can find more inner patterns in forums, because forums are a kind of conversation between users, therefore users quote often each other. On the other side, blog users can quote also, but they comment usually only the original blog post. Wikis do rarely have inner patterns.

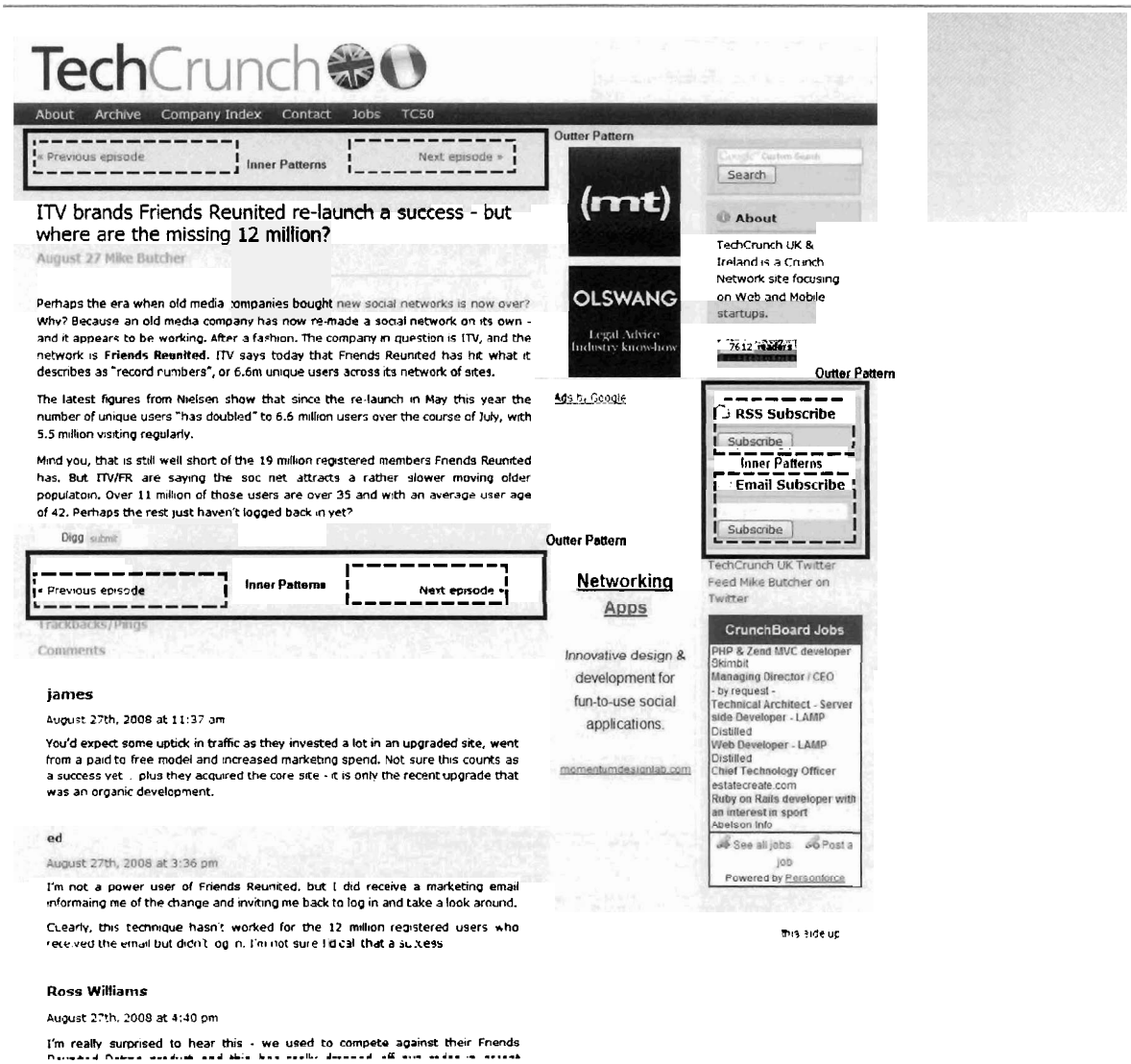

Figure 5: Screenshot of a blog's post

In order to exemplify this behaviour, let us take a blog entry in the TechCrunch Blog (fig. 5) and a post in the forum of the German journal "Spiegel" (fig. 6): Outer patterns are marked in black color and inner patterns in dotted lines. In the Blog site, we can only find four outer patterns, but in the forum page 26. In contrast to this, we find in our example for a wiki page zero outer patterns. Please note that a page having no outer patterns can still have patterns in it, but we only speak about outer patterns, if a pattern has at least two children patterns.

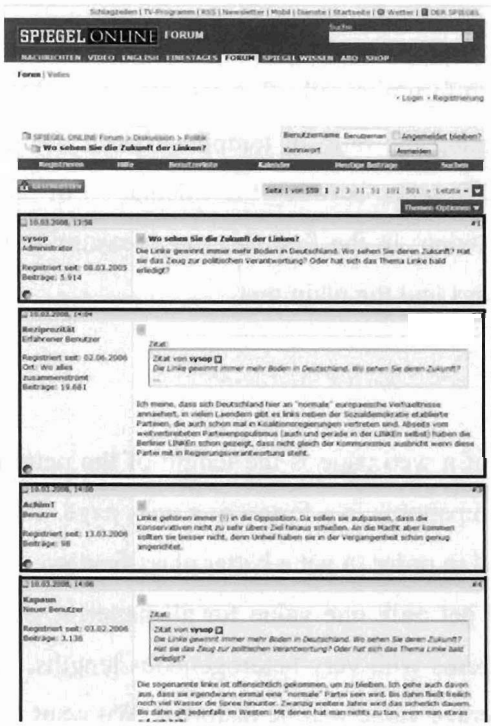

Figure 6: Screenshot of a forum's thread

3.3 Ratio of patterned vs. unpatterned Code

As we said before, blogs and forums are usually patterned on a more granular scope than wiki pages. This behaviour is reflected in the proportion of patterned and unpatterned code in them. We observed that a differentiation between the ratio of plain text and the ratio of effective HTML code can lead to a more accurate categorization of web pages. We want to clarify this fact in the next two subsections.

Ratio of templated vs. untemplated HTML code

Like already mentioned in section 3.1, we need more features in order to differentiate blogs and forums, thus we introduce the feature of ratio between templated and untemplated HTML code, which can be helpful for this cause. We observed that in a big number of forums, comments are usually smaller than comments in blogs. This results of the different function of blogs and forums: While forums are used to ask and solve different kind of problems that people have, like problems with a computer virus or a mother asking for a homemade medicine against common cold, blogs are more focused in a topic, that many persons have a different opinion about, e.g. global warming or politics.

In the first case, people want preferably short and simple solutions, but in the second case each user wants to explain in detail his point of view.

Because of this fact blogs have usually more HTML mark-up covered by patterns. Furthermore blogs have normally multiple sidebars displaying, e.g. general information about the author or the blog. Thus, we calculate the coverage of all patterns in the page. This value can be used as an additional value to differentiate web genres, because it should be bigger in blogs than in forums.

Ratio of templated vs. untemplated plain text

On the other side, blogs posts are usually longer than forum threads, thus we get often a bigger ratio of plain text in blogs than in forums. Hence, calculating the ratio of templated plain text can give us information about the genre of a document. The difference between the calculation of the ratio with HTML code and the ratio of plain text is the following: In the first case we measure only using the plain text and in the second case using the HTML tags and the plain text.

3.4 Length of patterns

Another important feature to determine the genre of a web page is the length of the patterns inside a document: Patterns in a blog are usually longer than patterns in a forum or a wiki page. The reason for this is a longer text in blog's posts. This can be used in order to get a better classification. We compute the average value of all pattern length in order to get only one value for all patterns. But there is a problem with this approach: If a page contains patterns with very heterogeneous lengths, for example many little patterns but one very long, then our average value will be distorted. We want to explain in the next subsections how we handle this distortion and how these values can help categorizing web pages.

Mean length of patterns

Computing the mean length of patterns is very easy, only the length values (in characters) are needed for this calculation. The average length is meaningful, if the values are not very heterogeneous, for example, a web page has four patterns with lengths [10, 20, 30, 40], the average value will be 25. This value is very representative for the example page. We said before that blogs post are usually longer than forums threads; this feature can be helpful in order to confirm this assumption.

On the other side, wiki pages are sometimes templated but contain very long passages of text. Problems arise if the lengths of the patterns are too heterogeneous, for example, we assume that a fifth pattern has the length 1000: In this case, we have [10, 20, 30, 40, 1000] and the average is 220. But 220 is not representative for these patterns, because 80 % of the patterns are smaller than 220. Hence, we decided to consider an additional value.

Median length of patterns

We define an additional value for better estimation of the length of patterns, the median length of patterns. The median length of patterns is not only specified by the values like the mean length of patterns but also with the number of patterns. To clarify this fact, we use two examples: These examples are based on two scenarios with different lengths [10, 20, 30, 40, 50] and [10, 20, 30, 40]. Note that in contrast to the mean length, the list for the calculation of the median length needs to be sorted. We have to examine two different cases:

1. If the number of elements is odd, like in the first example, then the median length is the element in the middle of the list, also in our first example 30.
2. Otherwise, the median length is defined as the average between the two values in the middle of the list. If our second example, the elements in the middle are 20 and 30, so is the median length 25.

Now we want to re-examine, the example mentioned in 3.4.1, where a length value was much bigger than all the others: [10, 20, 30, 40, 1000]. In this case is the median length 30 and this value is more representative for our patterns than the mean length (220). Thus we can see that the median length of patterns can be useful in order to describe length of patterns better.

3.5 Offset before first pattern starts

Let us take a look at our examples: The TechCrunch Blog (fig. 7) and the Spiegel forum (fig. 8). We observe that comments have a similar structure, so we can handle them as patterns. We proceed in the same way with forum entries and observe a similar structure too. We noticed that the most actual post is located at the top of blog pages. This post is usually longer than the offset between the begin of the page and the first pattern. We observe this behaviour in many pages, usually we have to scroll down to find similar structures in blogs but in forums this do not happen, patterns are close to the beginning of the page.

Figure 7: Screenshot of the first post in a blog

Following this observation we developed the feature *offset before first pattern starts*. This feature computes the offset between the begin of the document and the first pattern in the document. In blogs this value will usually be bigger than in forums. In wiki pages we do not find such an offset

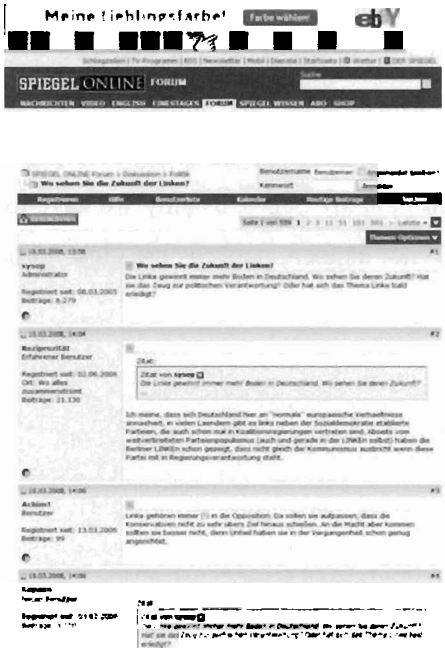

Figure 8: Screenshot of a forum with its head

3.6 Depth of patterns

The position or depth of a pattern can play an important role in order to detect if a page has a flat structure. A flat structure is often a characteristic of wiki pages; in contrast to wiki pages, blogs and forums are often table-based, so that their structure is deeper. This section explains how we can handle this information in order to determine the genre of a web page.

Mean depth of patterns

The first feature that we can extract from the depth of patterns is the average depth of all values. This value will be low for wiki pages and high for forums and blogs. For this feature we do not need to differentiate between outer or inner patterns, we just calculate the depth of each pattern and then compute the average of the different values.

Median depth of pattern

Like in Section 3.4, we differentiate between the mean and the median value. It is helpful in order to get a better representation of the depth in a document.

4 Evaluation

To evaluate our approach we need a corpus that provides all web genres that we want to classify. A corpus consists of a set of web pages that belongs to certain classes. There are some corpora available in the World Wide Web from previous works[7], but we had to build our own corpus, because other web genres were used in our approach and most corpora are outdated. In this section we want to explain, how we built our corpus, how many pages we collected for each genres and we want to present the result of some of our experiments.

4.1 Corpus

After an analysis of the chosen web genres (blogs, forums and wikis), we decided to split the genres blogs and forums into subgenres in order to deal with the structural diversity within the web genre. Blog and forum pages can be very heterogeneous in their structure in the same genre. For example, in the start page of the blog TechCrunch (fig. 2 and 7) and a post page (fig. 5) in the same blog we can see immediately the differences between the two pages: the first one gives an overview about the different posts in the blog and the second one is a post with many comments. While different posts have a similar structure, post's comments have a different structure. Thus we divided the blog genre into two different subgenres (Blog Start page and Blog Post Page).

In Forum pages we have the same case; there is a difference between the structure of start pages and thread pages, e.g. the Spiegel forum's start page (fig. 3 and 8) and a thread page (fig. 6) from the same forum. Also we divided the genre in the subgenres Forum Start Page and Forum Thread Page. For wiki pages it is not necessary to split the genre, because there is no significant difference between start page and arbitrary pages.

To build a good corpus is a difficult task, because automatically collection of web pages does not guarantee, that web pages have been labelled correct i.e. as a blog, forum or wiki page. We have to be sure that the web pages in the corpus were correct labelled, otherwise it is very difficult to train a classifier. In order to avoid manually collection of web pages, we tried to use compilations of web pages that are available from the World Wide Web and checked by other users.

We used the web directory dmoz[8] as a source for blog start pages. From this site we got 15000 different instances from different blog applications (wordpress, blogspot, etc) and languages. After that, we had to get single blog post pages from these instances, thus we extracted one link from each page (where possible automatically), resulting in 5900 blog post pages. For forums we followed the

---

[7] http://www.itri.brighton.ac.uk/~Marina.Santini/ [online 28.05.2008]

[8] http://www.dmoz.org/ [online 28.05.2008]

same procedure, but we used the web directory Big Boards[9], ensuring that different forums applications and forums in different languages were contained. We got 2000 instances and used them to obtain 1300 forum thread pages. We found Wiki pages' compilations in the WikiIndex[10] website and in Wikiservice[11] site, finally getting 3100 instances in different languages and from different applications.

4.2 Method of evaluation

Two things are needed in order to train a classifier for web genres: a corpus and a set of features. The corpus contains a set of correctly classified instances (also known as "learning set"), which are required from the classifier to learn. Good features should be distinctive. For our evaluation we used the Weka Machine Learning Toolkit [WF05]. Weka is a framework for machine learning providing a collection of algorithms for data mining tasks, but also contains tools for pre-processing, classification, regression, clustering, association rules, and visualization. Weka allows for example the evaluation of the different features in order to know which the most distinctive features are.

We selected a random subset of the corpus in order to have a uniformed corpus for training. We chose a total of 1005 instances (201 instances for each genre or subgenre). We applied different machine learning algorithms (Support vector machines with Sequential Minimal Optimization (SMO) [P99], C4.5-Decision Trees (J48) [Q93], Bayes Net [B04] and Random Forest Decision Trees [B01]) to our subcorpus in order to know, which algorithm performs best: All our classifications results were applied to 10-fold cross validation. A n-fold cross-validation is a statistical practice that consists of splitting a sample of data in into n subsets; one of the subsamples is retained for validation or testing and the other n-1 subsets are used to train the classifier. This process is called n-fold cross validation because this process is repeated n times (folds), so that each of the subsamples is used exactly once as validation data.

4.3 Results

We achieved over 62 % *accuracy* in our best results using the random forest classifier. Accuracy indicates the rate of correctly predicted instances. This is a good result, if we take to account, that we use only nine features. A random classification would produce an accuracy of 20 % (because we deal with five subgenres). By classifying instances to a class, there are four different cases to consider:

1. *True Positive* (TP): A classified instance is called TP, if it is labelled as belonging to the correct class.

---

[9] http://www.big-boards.com/ [online 28.05.2008]

[10] http://www.wikiindex.org/index.php?title=Category:All [online 28.05.2008]

[11] http://www.wikiservice.at/gruender/wiki.cgi?WikiVerzeichnis [online 28.05.2008]

2. *False Positive* (FP): An instance that is incorrectly labelled as belonging to the class.
3. *True Negative* (TN): An instance that is correctly classified as not belonging to the class.
4. *False Negative* (FN): An instance that is incorrectly classified as not belonging to the class.

We define accuracy as the rate of correctly predicted genres, i.e. the number of correctly predicted instances divided by the total number of instances. In table 1 you can see our best result with other statistical values like *precision*, *recall* and *F-Measure*: the precision for a class is the number of the system's correct predictions divided by the number of instances belonging to the class. Recall is the number of instances that have been correctly labelled as belonging to the class divided by the number of correct answers. F-Measure is a harmonic mean of recall and precision[12].

classified as →	a	b	c	d	e
a = Blog page	112	21	33	20	15
b = Wiki page	24	128	7	29	13
c = Forum page	18	9	147	9	18
d = Blog post	35	37	8	113	8
e = Forum thread	33	14	9	12	133
Precision	0.507	0.61	0.721	0.617	0.711
Recall	0.557	0.637	0.731	0.562	0.662
F-Measure	0.531	0.623	0.726	0.589	0.686
Accuracy	62 %				

Table 1: Confusion matrix and shortened example result of classification

On the higher part of fig. 9 we can see an example for a *confusion matrix*. A confusion matrix is a table that shows correctly and incorrectly classified instances. On the top of the figures we can see instances which were classified as a given class and on the right side real class of the instances.

Regarding our best results, we see that the only precision value under 60 % is the precision of the blog page class; for the other classes we get pretty good results, if we consider that classification by pure guessing has an accuracy of 20 % (taking into account 5 classes). The values for recall of blog pages and blog posts are under 60% too, but on the other side we get very good results for forum pages and forum threads.

We analyzed our structural features in order to rank the features by the information gained using this feature. This was done using the Information Gain Algorithm [M07] in WEKA. The result can be seen in fig. 10.

[12] http://en.wikipedia.org/wiki/Information_retrieval [online 03.06.2008]

Search Method: Attribute ranking	
Attribute Evaluator (supervised): Information Gain Raking Filter	
Ranked Attributes	
pattern_median_length	0.4634
pattern_ratio_text	0.4184
pattern_ratio_html	0.3646
pattern_nr	0.3642
pattern_mean_length	0.2888
pattern_start	0.2416
pattern_depth_mean	0.1492
pattern_outer	0.1003
pattern depth median	0.0796

Table 2: Shortened example result of feature evaluator

Ranking all features shows that two of our newly proposed features (`pattern_median_length` (e.g. see Section 3.4.2) and `pattern_ratio_text` (e.g. see Section 3.4.2) are especially important features, but also `pattern_ratio_html` (e.g. see Section 3.3.1) and `pattern_nr` (e.g. see Section 3.1). A page ranked with 0.4634 means, that 46 % of the pages could be correct classified with this feature.

We evaluated our approach using the Meyer zu Eissen corpus and their genres [SM04]. We achieved 28,25 % accuracy, but these result are not very good, because the result of Meyer zu Eissen ranged between 60% - 80%. Figure 11 shows a summary of our results.

classified as →	a	b	c	d	e	f	g	h
a = Articles	41	4	9	24	14	8	21	2
b = Discussion	8	61	11	4	14	12	6	11
c = Download	7	12	33	21	22	31	6	19
d = Help	30	11	21	21	18	15	12	11
e = Link lists	25	19	31	18	59	18	15	19
f = Portrait non-private	10	9	28	16	25	41	9	25
g = Portrait private	30	9	13	9	16	9	37	3
h = Shop	3	20	27	14	22	30	5	46
Precision	0.266	0.421	0.191	0.165	0.311	0.25	0.333	0.338
Recall	0.333	0.48	0.219	0.151	0.289	0.252	0.294	0.275
F-Measure	0.296	0.449	0.204	0.158	0.299	0.251	0.312	0.304
Accuracy	28.25 %							

Table 3: Confusion matrix and shortened example result of classification

We think that there are two more important reasons for these results are the identified genres and the age of the corpus. Some of their genres are too similar in their structure and differ only in the style of the language. "Non-private portrayals" and "private portrayals" have a same structure, but they were design for a different audience. On the other side, the corpus was built in 2004 and many pages do not exist anymore. The focus of the creation of web pages was in the presentation and not in the structure.

Conclusion

In this technical report we presented in detail a new type of structural features for detecting web genres. Structural features have not been applied to web genre detection before. Our results show that it is possible to achieve a reasonable accuracy for detection of structured web genres like blogs, forums and wiki pages. Previous approaches use features like number of tags, intern and extern links only. We focus on patterns, which are structural fragments of a web page that are re-occurring in the page's HTML.

Our approach uses the HTML structure of a document in order to classify it. The first step of our approach consists in finding all patterns in a web page. Having the patterns of a page, we use attributes of these patterns like the length or the number of patterns as features. For the evaluation, we used machine learning techniques built a corpus of more than 33.000 instances and used 1005 random chosen instances for training and test purposes.

We conclude that it is possible to classify web pages without knowing the content's language. Other approaches can lead to better results, but we obtained our results using a small set of features (9 features). This is because we avoid the use of linguistic features. Approaches using linguistic features often have up to 1033 features [S04a].

Our proposed structural features were not very distinctive alone, but they can contribute in order to get a better classification of web pages. Based on these results we can use our structural features in order to complement the known features from literature. But this is still a research question that has to be handled in future work.

References

[B01] L. Breiman. Random Forest. Machine Learning, Volume 45, # 1, pp. 5-32, 2001.

[B04] R. Bouckaert. Bayesian networks in Weka. Technical Report 14/2004. Computer Science Department. University of Waikato. 2004.

[BH05a] E. Boese and A. Howe. Genre Classification of Web Documents. Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05). Poster paper, Pittsburgh, Pennsylvania (USA), 2005.

[BH05b] E. Boese and A. Howe. Effects of Web Document Evolution on Genre Classification. In Proceedings of the 14th ACM international conference on information and knowledge management (CIKM '05), ACM, pp. 632-639, 2005.

[DVM01] N. Dewdney, C. Van Ess-Dykema and R. MacMillan. The Form is the Substance: Classification of Genres in Text. In Proceedings of the workshop on Human Language Technology and Knowledge Management, pp. 1-8, 2001.

[F48] R. Flesch. A new readability yardstick. Journal of Applied Psychology, 32:221-233, 1948.

[FB06] G. Friezes and P. Bailey. Towards Practical Genre Classification of Web Document. In Proceedings of the 15th international conference on World Wide Web, pp. 1013-1014, 2006.

[FK06] A. Finn and N. Kushmerick. Learning to Classify Documents According to Genre. In Journal of the American Society for Information Science and Technology (JASIST), Special Issue on Computational Analysis of Style, Volume 7, pp. 99.3, 2006.

[GZZCK05] P. Guo, Y. Zhou, J. Zhuang, T. Chen and Y. Kang. An Efficient Algorithm for Mining Both Closed and Maximal Frequent Free Subtrees Using Canonical Forms. Lecture Notes in Computer Science Volume 3584. Advanced Data Mining and Applications, pp. 96-107, ISBN 978-3-540-27894-8, 2005.

[JWZ95] T. Jiang, L. Wang and K. Zhang. Alignment of trees – an alternative to tree edit. Theoretical Computer Science, Vol. 143, # 1, pp. 137-148, 1995.

[KC99] M. Kim and K. Choi. A comparison of collocation-based similarity measures in query expansion. Information Processing and Management: an International Journal, Volume 35, # 1, pp. 19-30, ISSN: 0303-4573, 1999.

[KLE98] P. N. Klein. Computing the Edit-Distance Between Unrooted Ordered Trees. Lectures Notes in Computer Science, Volume 1461, pp. 91-102, ISBN: 978-3-540-64848-8_8, 1998.

[KNS97] B. Kessler, G. Numberg and H. Schütze. Automatic detection of text genre. In Proceedings of the 35th Annual Meeting on Association for Computational Linguistics, pp. 32-38, 1997.

[KNM05] D. Katsaros, A. Nanopoulos and Y. Monolopoulos. Fast mining of Frequent Tree Structures by Hashing and Indexing. Information and Software Technology, ISSN 0950-5849, Vol. 47, # 2, pp. 129-140, 2005.

[L06] A. Lahn. Genre-Analyse von Web-Dokumenten. Master Thesis. Bauhaus-Universität Weimar, 2006.

[LLK04] C. Lim, K. Lee and G. Kim. Automatic Genre Detection of Web Documents. Natural Language Processing – IJCNLP 2004, Springer Berlin / Heidelberg, LNAI, pp. 310-319, 2004.

[M97] T. Mitchell. Machine Learning. McGraw-Hill International Editions, Computer Science Serie, pp. 57-59, ISBN 0-07-115467-1, 1997.

[NKTM00] K.Nigam, A. Kachites McCallum, S. Thrum and T. Mitchell. Text Classification from Labeled and Unlabeled Documents Using EM. Machine Learning, 39 (2-3):103-134, 2000.

[PGW95] Y. Papakonstantinou, H. Garcia-Molina and J. Wisdom. Object exchange across heterogeneous information sources. In Proceedings of the IEEE International Conference on Data Engineering (ICDE), pp. 251-260, 1995.

[P99] J. Platt. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schoelkopf and C. Burges and A. Smola, editors, Advances in Kernel Methods – Support Vector Learning, Chapter 12, pp. 185-208, MIT Press, 1999.

[Q93] R. Quilan. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[RMS06] D. Rafiei, D. Moise and D. Sun. Finding Syntactic Similarities Between XML Documents. In Proceedings of the Conference on database and Expert Systems Applications (DEXA), IEEE Computer Society, pp. 512-516.

[S04a] M. Santini and A Shallow. Approach To Syntactic Feature Extraction for genre Classification. In proceedings of the 7th Annual Colloquium for the UK Special Interest Group for Computational Linguistics, 2004.

[S04b] M. Santini. State-of-the-Art on Automatic Genre Identification. University of Brighton, Technical Report, ITRI-04-03, 2004.

[S06a] M. Santini. Description of 3 Feature Sets for Automatic Identification of Genres in Web Pages, 2006.

http://www.nltg.brighton.ac.uk/home/Marina.Santini/three_feature_sets.pdf [21/05/2008]

[S06b] M. Santini. Some Issues in Automatic Genre Classification of Web Pages. JADT 06 – Actes des 8 Journées internationals d'analyse statistiques des donnés textuelles, Vol 2, 865-876, 2006.

[S07] M. Santini. Characterizing Genres of Web Pages: Genre Hybridism and Individualization. On Proceedings of the 40th Hawaii International Conference on System Sciences, 2007.

[SM04] S. Meyer zu Eissen and B. Stein. Genre Classification of Web Pages: User Study and Feasibility Analysis. In proceedings of 27th German Conference on Artificial Intelligence (KI 2004), pp. 256-269, Springer LNAI 3228 (2004), 2004.

[SM06] B. Stein and S. Meyer zu Eissen. Is Web Genre Identification Feasible?. 17th European Conference on Artificial Intelligence (ECAI 06) Brewka, Coradeschi, Perini, Traverso (Eds.) pp. 815-816, ISBN 1-58603-642-4, IOS Press 2006

[TWA06] M. Tsukada, T. Washio and H. Motoda. Automatic Web-Page Classification by Using Machine Learning Methods. Lecture Notes in Computer Science, Vol. 2198. Proceedings of the First Asia-Pacific Conference on Web Intelligence: Research and Development, pp. 303 – 313, ISBN 3-540-42730-9, 2001

[VLG07] V. Vidulin, M. Lustrek and M. Gams. Training a Genre Classifier for Automatic Classification of Web, Journal of Computing and Information Technology – CIT 15, 2007, 4, 305 – 311, doi: 10.2498/cit. 1001137, 2007.

[WL00] K. Wang and H. Liu. Discovering structural association of semistructured data. IEEE Transactions on Knowledge and Data Engineering 12 (3), pp. 353-371, 2000.

[WF05] I. Witten and E. Frank. Data Mining: Practical machine learning tools and techniques, 2nd edition, Morgan Kaufmann, San Francisco, 2005.

[ZSS92] K. Zhang, R. Statman and D. Shasha. On the editing distance between unordered labelled trees. Information Processing Letters, 42(3):133-139, 1992.