Fast and Scalable Pattern Mining for Media-Type Focused Crawling*

[experience paper]

Jürgen Umbrich and **Marcel Karnstedt** and **Andreas Harth**†
Digital Enterprise Research Institute (DERI)
National University of Ireland, Galway, Ireland
firstname.lastname@deri.org

Abstract

Search engines targeting content other than hypertext documents require a crawler that discovers resources identifying files of certain media types. Naïve crawling approaches do not guarantee a sufficient supply of new URIs (Uniform Resource Identifiers) to visit; effective and scalable mechanisms for discovering and crawling targeted resources are needed. One promising approach is to use data mining techniques to identify the media type of a resource without the need for downloading the content of the resource. The idea is to use a learning approach on features derived from patterns occuring in the resource identifier. We present a focused crawler as a use case for fast and scalable data mining and discuss classification and pattern mining techniques suited for selecting resources satisfying specified media types. We show that we can process an average of 17,000 URIs/second and still detect the media type of resources with a precision of more than 80% and a recall of over 65% for all media types.

1 Introduction

The number of search engines focused on specific topics has increased significantly over recent years. Besides search engines focused on (hyper)text documents, specialised search engines are available online which collect and integrate information from files of particular media types. Seeqpod [URL, j] and Blinkx [URL, g] offer search over audio and video files, Google Scholar [URL, b] and CiteSeer [URL, a] are digital libraries of printable documents, Technorati [URL, d] provides real-time access to news-feeds, and Seekda [URL, c] offers search capabilities for web services. A common issue for all vertical search engines is the challenge of discovering and downloading the targeted files on the Web. Specifically, the challenge of detecting documents of a certain media type without inspecting the content is still not solved [Lausen and Haselwanter, 2007]. For this task, a URI-only classifier is a good choice, because speed is crucial and content filtering should be enabled before an (objectionable) web page is downloaded. Basically, a focused crawler ([Chakrabarti *et al.*, 1999]) wants to infer the topic of a target page before devoting bandwidth to download it. Further, a page's content may be hidden in images.

A crawler for media type targeted search engines is focused on the document formats (such as audio and video) instead of the topic covered by the documents. For a scalable media type focused crawler it is absolutely essential to discover documents of the requested media type on the Web and to avoid expensive HTTP lookups of irrelevant files. Thus, the crawler needs to identify the media type of a document without establishing a connection and downloading the content. A common way to identify the format of a file is to use the file extension of the file name or to detect characteristic byte patterns in the file content itself (magic number approach), which does not scale well. The latter approach is not suitable because it requires to retrieve the data which is expensive and time consuming task. We can conclude that the file extension is only for some media types suitable as an identifier based on a study of 22M Web documents [Umbrich *et al.*, 2008] in 2008.

We propose to use classification or pattern mining techniques to discover Web documents of requested media types without analysing the content. For this, we utilise information available for a crawler during the crawling process to classify and filter URIs for pre-defined media types. Note that learning patterns without analysing the content of files can be applied in other scenarios as well, e.g., in genre classification. We present general data mining approaches suited for this task, discuss their strengths and weaknesses, and, based on first experiences, present a classifier-based solution. As this method still comes with several disadvantages, we further propose to apply frequent pattern mining approaches as an alternative. Here we focus on stream mining approaches, as they provide a fast and scalable solution that is perfectly suited for the long-running and input-intensive task of a Web crawler.

The remainder of the paper is organised as follows: In Section 2 we briefly present the basics of a crawler focusing on media-types and discuss general data mining approaches suited for this. Section 3 presents first experiences that we gained using implementations available in the WEKA toolkit. Based on these experiences, we propose a classifier approach in Section 4 and discuss an improvement based on pattern stream mining. Section 5 contains an evaluation of the classifier approach. Finally, Section 6 briefly presents related work and Section 7 concludes the paper.

*This material is based upon works jointly supported by the Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2) and under Grant No. 08/SRC/I1407 (Clique: Graph & Network Analysis Cluster)

†Current affiliation: Institut AIFB, Universität Karlsruhe (TH)

2 Data Mining for Focused Crawling

The principle of a crawler is to start with a set of seed URIs and to recursively follow the discovered links. For this, crawlers follow, for instance, a breadth-first or depth-first approach. This means, the crawler changes domains through the crawling job and may be returning to domains already visited before. A focused crawler tries to follow only links to pages and files of a specific type – in our case the media type of files.

Our basic idea is to utilise the information available for a crawling during the crawl loop to identify the media-type of Web resources without inspecting the content itself. The accessible information sources for a crawler are [Umbrich *et al.*, 2008]:

1. the **URI**
2. the **link position** of the URI in the HTML document
3. the information sent with the **HTTP response header**

However, in this work we exclusively focus only on the mining of features contained in the URI. URIs are also used for topic classification [Baykan *et al.*, 2009]. Media types are registered with their related RFC description with IANA [URL, h]. The RFC description contains a general explanation of the purpose of the media type and also recommendation for the file extension(s) to use when publishing a document. A media type consists of two parts, 1) the `content-type` and 2) the `sub-type`, separated by "/" (`content-type`/`sub-type`). To explain our thoughts we introduce an example URI[1] referring to a W3C Video on the Web Workshop in 2007.

2.1 URI

Every network-retrievable document has a Uniform Resource Identifier (URI) as defined in RFC2396 [URL, i], which specifies that a URI consists of five components.

[*PROTOCOL*]://[*HOST*]:[*PORT*]/[*PATH*][*FILE*]?[*QUERY*]

Please note that we focus exclusively on the Hypertext Transfer Protocol (HTTP) and omit the other protocols, such as the File Transfer Protocol (FTP, RFC 959 [URL, f]) or the gopher protocol (RFC 1436 [URL, e]). The single parts have the following meanings:

- The *PROTOCOL* specifies the transfer protocol used to access the resource on the Internet.
- The *HOST* part refers to the domain name of the web server.
- The *PORT* component depends on factors such as server-side firewall settings, proxy configuration or router settings. If the *PORT* component is omitted, the standard assumes the default HTTP port 80.
- The *PATH* and *FILE* components can be created either automatically, for example by a content management system, or manually by human users.

The *PATH* component can be part of a hierarchical folder structure. Users organise their content in a folder structure, such as storing images in an `image` directory or videos in a `video` sub folder. In this case, special sub folders can be used as an indicator for the media types of the documents in this folder. Our URI example has the *FILE* component `Angio.avi` with file extension `avi`, which is mapped to the media type `video/x-msvideo`. The mapped media type matches with the real media type of our example. The *QUERY* component is a string of information to be interpreted by the resource and is omitted here. We use the following notation for the URI components in this paper:

- **E** for the file extension of the *FILE* component
- **F** for the filename
- the single tokens in the *PATH* components are labeled with **T**
- the *HOST* part is labeled with **D**

2.2 Classification

We can map our media-type identification problem to a classification task. Based on a set of predefined media types, we want to classify a URI to its real media type. Therefore, we have to split the components of a URI into a feature set that serves as the input for the classification algorithm. Possible classification algorithms for this task could be one of the following: The **Bayesian classifier** shows convincing results for classifying text. Most spam detection applications use a Bayesian classifier to decide whether or not a text snippet or an email is spam[2] [Sahami *et al.*, 1998]. The **support vector machines** (SVM) algorithm is also used for focused web crawling [Sizov *et al.*, 2002] and achieved reasonably good results in classifying documents to topics. The C4.5. algorithm generates a pruned or **unpruned decision tree** [Quinlan, 1993] [Quinlan, 1996]. A common **Bayesian network** classification algorithm [Pearl, 1985] is also featured. Noteworthy, it is proven that a classification algorithm based on decision trees and word tokens from anchor text can significantly improve focused crawling [Li *et al.*, 2005]. Hence, we do not know how it performs for the classification task based on a small set of features.

Some disadvantages come along with the classification approach. If we use a supervised algorithm we need to generate a training set. Given that there exist over 200 different media types on the Web and that 80% of the documents are text/html documents, we have the problem of gathering a training set with representative candidates. Another important fact is the dynamic and heterogeneous nature of the Web. Different domains use different URI patterns for the same media type documents. The classifier could be optimised especially for some certain domains, depending on the training set of selected URIs. Such a domain specific classifier could achieve a very good classification precision for certain URIs, but the classifier will "fail" for URIs of domains that are not in the training set.

This static approach is not suitable without retraining the classifier during the crawl. This is especially an issue in the context of changing domains, where already visited domains may be visited later on again. Also, the training and testing tasks could be very time consuming in the context the Web crawling application. We will show that in the next section.

2.3 Dynamic Pattern Mining

A very promising approach is the mining of patterns from URIs to identify the media type of the underlying document. Especially stream pattern mining seems to be a good solution. We will discuss the stream mining approach in more detail in Section 4. The basic idea is to mine frequent

[1] http://www.w3.org/2007/08/video/slides/KidsHealth/Angio.avi

[2] http://www.paulgraham.com/Spam.html

Content type	Classifier	Max precision	Features
application/*	j48	95.24%	DTFE
audio/*	Bayes	94.09%	TE
image/*	smo	100.00%	E
text/*	smo	91.03%	DTFE
video/*	Bayesnet	56.96%	TE

Table 1: Selected results for the best classifier & feature combination with the highest precision for each tested content types.

patterns from a set of URIs and their real media types. Based on these association patterns we can learn rules and use them to identify the media types. With pattern stream mining approaches we can discover new patterns over the time the crawler traverses the Web. This approach can also keep up with new appearing media types and does not rely on a predefined and static set of media types.

3 First Experiences

First, we wanted to know if a classification approach is suited for our needs at all. We used the WEKA toolkit to gain first knowledge if patterns extracted from URIs are suitable for a media type classification. Based on the possible features of a URI, we tested and compared four different classification algorithms provided by the WEKA framework. The evaluation contains the results of all possible permutations of classifier and feature combination for media types and content types. However, we present here only selected results out of 19 test runs. The evaluation shows precision and recall values as well as detailed results of the time needed to train and test the classifiers. Further, we studied the scalability of the WEKA library and the provided implementations.

3.1 Feature Combinations

The test for possible feature combinations contains the following four feature combinations: **E**, **FE**, **TFE**, **DTFE**. They were chosen intuitively in order to get a first impression of the suitability of the approach. We observed that there exists no clear feature combination that in general achieves the highest precision and recall values. In 9 out of 19 tests the feature combination DTFE (domain, tokens, file name and file extension) achieved the best precision. Another objective fact is that there exists no single classification algorithm that clearly outperforms the others. Table 1 shows the summary of the classifiers that have the highest precision to identify documents for the tested media types. 12 out of 19 classifier algorithms uses the Bayesian theory. The Bayes classifier achieves in 8 out of 19 cases the best precision value and the Bayesian net algorithm in another 4 cases. The complete list of the results[3] and more details are provided in [Umbrich, 2008].

3.2 Scalability

We stopped the execution time to train and test a certain WEKA classifier after a number of input instances. The benchmarks are performed for all possible combinations of classifiers and feature patterns for the content type `application` with a dictionary containing 5,000 words. The results are for the feature combination DFTE. Figure 1 shows the time to generate the WEKA input (ARFF) file,

[3] http://www.umbrich.net/pubs/master_thesis.pdf

Figure 2 shows the result to train the algorithms and Figure 3 the results to test the trained classifiers. The major problem and scalability limitation of the WEKA workbench is that all the information that is needed to train a classifier and finally to classify new instances are kept in memory. Thus, the limitation of WEKA is the available inmemory space. With the naive Bayes classifier the memory limit was reached with a dictionary containing 20,000 tokens. Another reason why we believe that WEKA is not suitable for a scalable architecture is the bad time performance to classify a list of URIs. Results show that the average classification speed is 47.5 URIs/seconds. Under the assumption that the system filters millions of URIs, the current implementation with the WEKA libraries is not applicable. The time to create the required ARFF file increases exponentially with the number of instances, as the results in Figure 1 show.

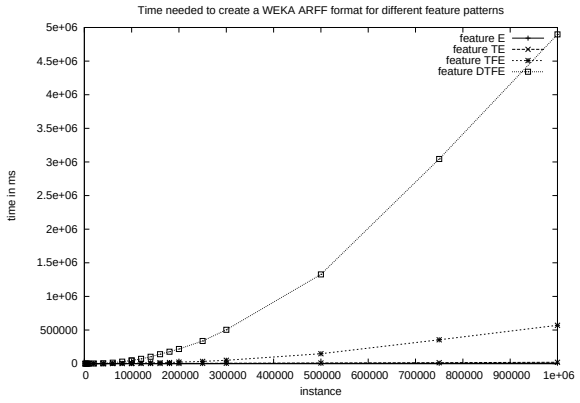

Figure 1: Elapsed time to generate the WEKA ARFF input format.

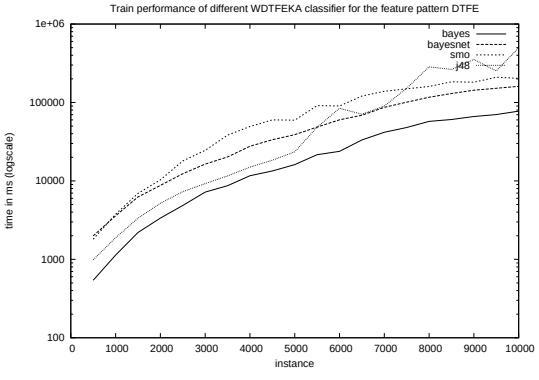

Figure 2: Elapsed time to train a WEKA classifier for each feature combination.

The conclusion of this first evaluation is that the provided JAVA implementations of WEKA have memory limitations. For our Web crawler application, the classifiers cannot keep up with the performance of URIs/sec supplied by the crawler.

4 Approach

From the tested classification algorithms, the Bayesian classifier was the most suitable, in terms of used mem-

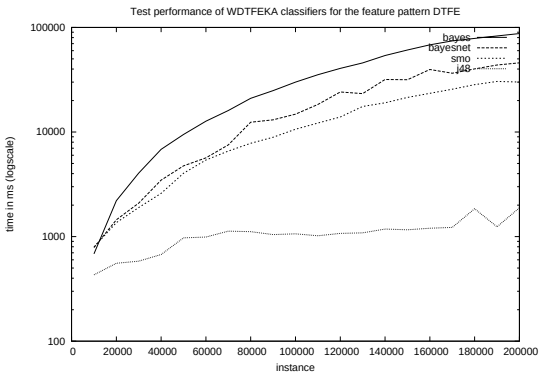

Figure 3: Elapsed time to test a WEKA classifier for each feature combination.

ory and the performance of the learning and classification task. Thus, we propose a statistical classifier similar to the *a priori* approach of the Bayesian algorithm or "One-Item-Rule" of an association rule algorithm [Agrawal *et al.*, 1993].

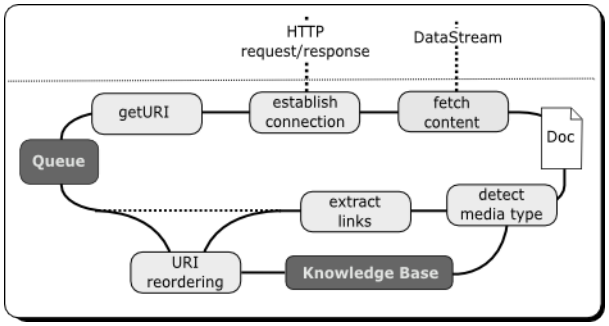

Figure 4: Media type crawl filter architecture

The general idea is illustrated in Figure 4. A knowledge base holds the data to generate files required to build and evaluate new classifiers. Furthermore, the knowledge base analyses the inserted information and generates meta data about, for example, the distribution of media types, the occurrence of feature values and the appearance of features together with media types. These meta data are used for a statistical approach to determine the media type of a URI.

An evaluation function calculates a conditional probability of being a media type mt, given the features extracted from URI uri. In the current version, we implemented a media type filter that supports a statistical classifier. This is an algorithm similar to association rule mining, which is fed by background knowledge from the meta data of the data store component. Next, we will describe the algorithm of the proposed statistical classifier in detail.

4.1 Statistical Classifier

The statistical classifier calculates the relevance score based on the conditional probability with the function of the Bayesian theory,

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{|A \cap B|}{|B|}.$$

The classifier scans the meta information of the data store to select the best features to identify the media types and stores the selected features in an internal cache. The most precise features are selected with an algorithm similar to the *a priori* association rule mining algorithm [Agrawal *et al.*, 1993; Agrawal and Srikant, 1994]. The "One-Item-Rule"($A \Rightarrow B$) algorithm selects single features which are associated with a "goal" or class like $.jpg \Rightarrow image/jpeg$. The best features are selected based on the confidence value

$$conf(a \rightarrow b) = \frac{sup(a \cup b)}{sup(a)}$$

with

$$sup(x) = \frac{\#x}{\#total}.$$

The formula for calculating the confidence is equivalent to the conditional probability formula, which calculates the probability of some event A, given the occurrence of some other event B. Conditional probability is written

$$P(A|B) = \frac{P(A \cup B)}{P(B)},$$

and is read "the probability of A, given B".

The algorithm is not a full association rule algorithm and skips the rule detection for all possible feature combination. Instead, our algorithm uses the mapping function

$$\Pi(uri, featurePattern) = featureString$$

to generate all possible combination of features for given and pre-defined combination patterns.

A combination pattern is a series of single features, such as the combinations of **DT**, that is, all possible combination of the pair <pld+token>.

We use a true Bayesian estimate to calculate the importance of a rule based on the value of the conditional probability and the occurrence of the feature in our data store.

Bayesian Estimate The formula for calculating the weighted score for a "feature rule" gives a true Bayesian estimate:

$$wfeature = \frac{v}{v+m} \times R + \frac{m}{v+m} \times C$$

where R denotes probability for the feature, v the number of total occurrence of the feature, m the minimum occurrence required to be added to the rule set and C the default probability. This equation is adapted from the weighted rank of movies at IMDB[4].

The media type filter is the core component of our crawling framework to guide the crawler while traversing the Web for new documents of requested media types. With this component, the framework can perform a crawling strategy which is equal to a dynamical heuristic search algorithm. The media type filter integrates the mapping function $\Pi(uri)$ with an evaluation function $f(uri, \Pi(uri))$. The background knowledge used to calculate the relevance score of URIs can be updated during the crawl runtime and thus, the filter can learn new features to discover relevant documents. The mapping function $\Pi(uri) = \{TLD(uri), T(uri), N(uri), E(uri)\}$ extracts features from a URI and generates a feature vector that is used as input for the evaluation function. The evaluation function returns the conditional probability that a URI

[4]http://www.imdb.com/

is of media type mt, or expressed in terms of the probability theory $f(mt, \Pi(uri)) = P(mt|uri)$. With the relevance score from the evaluation function, the component filters a list of URIs and adds selected URIs to a single priority queue or a set of priority queues. The background knowledge of this component contains pairs of URIs with their real media types. The post-processing component detects the real media type of downloaded files and adds this knowledge into the media-type filter.

Filter rules control which pages are visited next and which URIs are ignored. The crawling behavior can be controlled with the following rules:

- $(mt, min\_prob) \Rightarrow ALLOW$ Allow/Select URIs with a probability of more than min_prob for media type mt.
- $(mt, min\_prob) \Rightarrow DISALLOW$ Disallow/Filter out URIs with a probability of more than min_prob for media type mt.

We are aware that the presented approach of a statistical classifier does not allow to easily change patterns or learn new media types. Obviously, an unsupervised approach that can mine the occurring patterns would be more practical. Hence, we discuss a pattern mining approach in the next section.

4.2 Pattern Tree Approaches

As we highlighted, the **limitations** of the presented approach are:

1. feature patterns are static and pre-defined
2. more feature pattern combinations require more in-memory space
3. new discovered media types cannot be added to the classifier

We found a promising approach using frequent pattern trees [Han *et al.*, 2000] and an extension to mine frequent itemsets in streams under dynamically changing resource constraints [Franke *et al.*, 2006]. Especially, the latter work fulfills our requirements. Franke et. al provide a framework to mine frequent itemsets, either from fixed size intervals or from time intervals. The algorithm is resource-aware, which means it can be adjusted to changing resources and it can be run with fixed allocated resources. This approach has the following **advantages**:

1. automatic detection of new patterns
2. assurance of a constant memory footprint independent from the crawling time or the number of processed URIs.
3. new discovered media types can be added to the classifier

Using a stream-based approach promises that the performance requirements of a Web crawler can be met without problems. The crawler produces a high input rate for the mining task (up to thousands of inserts per minute) and runs for days to weeks. Thus, we are running continuous mining queries on a continuous stream of high input rate. That is exactly what stream mining algorithms are developed for, as they usually aim for needing only one look on each date. Another advantage is the possibility to query for frequent itemsets from certain time intervals. If the crawler records at which time he enters or leaves a domain, it is straightforward to determine patterns from only that domain. In future work we plan to evaluate what is the better choice, either use all patterns learned so far or just focus on domain-specific ones. The algorithm from [Franke *et al.*, 2006] provides a perfect basis for that, which is accompanied by its resource awareness. Moreover, the algorithm can always provide guarantees on the achieved quality, depending on the currently used resources.

Meanwhile, we implemented the pattern mining approach. The next steps are to fully integrate it into the crawling framework. When this is done, we can measure the overall performance of this method and compare it to the static classifier approach. We expect similar performance, due to the stream character of the used solution. Concerning accuracy and applicability, we expect even better results, due to the dynamic and unsupervised features of the approach.

5 Evaluation

In this section, we present the methods and results of our evaluation of the statistical classifier. We measured the precision and recall of our implementation and further, the performance with respect to the requirements of a Web crawler use-case. We measured the discovery ratio of requested media types on the Web based on a base-line crawl with a breadth-first crawling strategy. First, we present the evaluation setup and used methods, followed by the results of the single evaluations steps. Finally, we provide a detailed discussion of the results.

5.1 Setup

All experiments are performed on a single Opteron 2.2 GHz CPU with 4 GB of main memory and two SATA (160GB,750GB) disks. We used a published and representative web corpus containing 22.2M documents [Umbrich *et al.*, 2008]. The test corpus contains 3.8M external links from ODP[5] and 6.4m external links from Wikipedia[6] in December 2007.

We use data from this combined corpus to evaluate the approach of our statistical classifier. We focus on the DTFE feature combination, as this has been proven to be the generally most suited one in our WEKA tests. The underlying assumption for this test is that as soon as the conditional probability of a feature is greater than 0.5 (=50%), we can assume a positive match. The classifier evaluates first the file extension of the URI, second the single path tokens and third, the combination of the top-level-domain and the path tokens. If all features cause a relevance value of less than 0.5, we will assume a false match.

Our test set contains 4.28 M detected media types and is split into eight folds, fold `A`, `B`, `C`, `D`, `E`, `F`, `G` and `H`. The last fold (H with 285396 entries) is used to evaluate and test the classifier. The first seven folds (`A`–`G`), containing 500.000 entries, are used to train and update the statistical classifier. We ran seven evaluation rounds for each of the five content types (`application`, `audio`, `image`, `text` and `video`). The classifier is updated by one of the remaining data set parts and tested against the separate data set `H` in each round. The internal cache of the classifier was set up with 5000 entries for each feature and with the thresholds 0.5 and 0.7. All features that have a conditional probability less than the threshold are deleted.

[5] http://rdf.dmoz.org/rdf/content.rdf.u8.gz

[6] http://download.wikimedia.org/enwiki/

5.2 Results

First, we list the precision and recall values from the evaluation. Second, we show the benchmarks to mine and classify new URIs. Table 2 list the precision and recall values for two thresholds and the five content types.

	Rounds						
	a	b	c	d	e	f	g
`application/*`							
Threshold 0.5							
Pr	85.24	85.34	86.21	85.93	86.05	85.84	85.67
Re	66.56	67.37	67.54	67.68	67.50	67.46	67.47
Threshold 0.7							
Pr:	91.98	91.96	93.34	93.10	93.22	93.04	93.13
Re:	66.50	68.44	66.48	65.38	65.34	65.32	65.32
`audio/*`							
Threshold 0.5							
Pr:	87.88	86.14	86.57	86.57	84.88	86.57	84.69
Re:	84.88	84.88	84.88	84.88	84.88	84.88	86.34
Threshold 0.7							
Pr:	91.58	91.58	91.10	91.10	91.10	91.10	91.10
Re:	84.88	84.88	84.88	84.88	84.88	84.88	84.88
`image/*`							
Threshold 0.5							
Pr:	84.89	84.05	83.50	82.96	83.37	83.53	83.23
Re:	98.31	98.43	98.64	98.64	98.66	98.66	98.68
Threshold 0.7							
Pr:	87.84	87.81	87.90	87.74	87.89	87.60	87.66
Re:	98.11	98.25	98.37	98.46	98.52	98.50	98.50
`text/*`							
Threshold 0.5							
Pr:	73.68	73.78	73.81	74.00	73.94	73.97	73.99
Re:	99.78	99.77	99.74	99.73	99.74	99.75	99.74
Threshold 0.7							
Pr:	86.74	87.26	87.47	88.85	88.71	88.77	88.83
Re:	61.36	61.07	61.01	60.06	60.16	59.91	59.90
`video/*`							
Threshold 0.5							
Pr:	72.73	72.73	71.11	72.73	71.11	71.11	71.11
Re:	91.43	91.43	91.43	91.43	91.43	91.43	91.43
Threshold 0.7							
Pr:	81.48	81.48	74.42	81.48	74.42	74.42	74.42
Re:	62.86	62.86	91.43	62.86	91.43	91.43	91.43

all values are percentages.

Table 2: Precision and recall of the statistical classifier for different thresholds.

Scalability We benchmarked the implementation of the statistical classifier to measure the time to insert/update and classify a list of URIs. Figure 5 shows results. Our implementation shows a linear increase of the elapsed time with the number of URIs for both operations.

Focused Crawling for Media Types We evaluate the applicability of our approach in a media-type focused Web crawler based on the following use case: Documents of content type `audio`, `video` or `image` are requested to build a multimedia search engine. The detailed setup of this experiment is as follows: We started with a seed set of twelve URIs, collected from the social bookmark page Delicious[7]. The requested documents cannot be used to extract new links, thus we selected also links to `text/html` documents. To measure the efficiency of the pattern mining approach we compared three typical crawling strategies. As our base-line we performed a breadth-first crawl. The second strategy is a focused crawl (best-first) using the statistical classifier without a cut-off threshold to omit

[7] http://delicious.com/

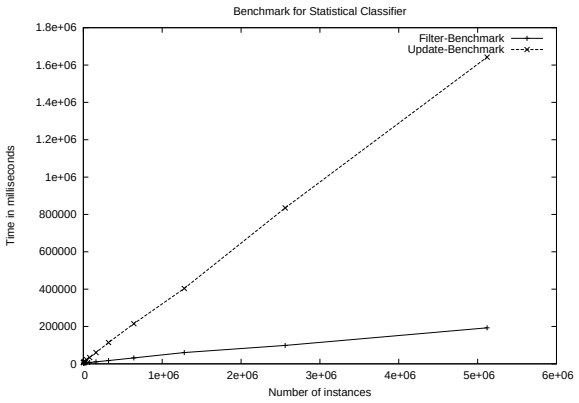

Figure 5: Performance of the statistical classifier.

URIs. We prioritised URIs based on their conditional probability that they are of a requested media type. The filter methods first selects all URIs which are of content type `audio`,`video` or `image` and adds the selected URIs to the queue. In a second processing step URIs of media type `text/html` are selected and forwarded to the queue. The third strategy applies a fixed size queue of 10K URIs and a URI-per-domain limit of 10. The results of the different strategies are presented in Figure 6.

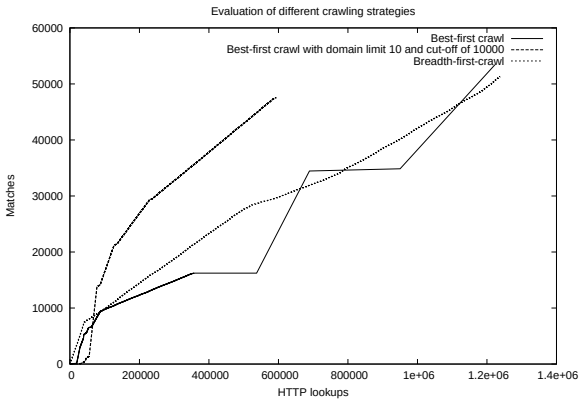

Figure 6: Performance of different crawl strategies.

5.3 Discussion

Next, we will discuss the results presented in the previous section. Because of space limitations we refer to a more detailed discussion of the results in [Umbrich, 2008].

Precision and Recall Table 2 lists the detailed results for the classification evaluation. First of all, we can observe that for all tested `content-types` the precision values are higher than 75% and the recall values above 60% . The predefined feature patterns achieve reasonable good results. However, we observe that the recall values for `video/*` and threshold 0.7 differ between 60% and 90%. This difference is caused by the pre-defined and static feature pat-

terns and boosts our idea to use the frequent pattern tree approach.

Furthermore, we observe an increase of the precision by nearly 9% if the internal cache threshold of the classifier is changed from 0.5 (= 50%) to 0.7 (=70%). To revive, features with a conditional probability less than the internal cache threshold for a certain content or media type are not recorded and used for the classification. With this threshold we can control the characteristics of the statistical classifier. A higher threshold results in a higher precision and a lower threshold increases the recall.

Comparing the precision of the statistical classifier with the achieved classification precisions of the WEKA classifiers we notice very promising results. Beside the precision values for the classification of documents of content type `image`, the statistical classification approach achieves precision values that are only around 2% less than the WEKA classifiers, with still reasonable good recall values for the content types `application`, `audio`, `text` and `video`.

As expected, the classifier approach achieves very good values for precision and recall. This shows that the approach of choosing data mining techniques to identify media types without checking file content is very suitable and applicable. The next interesting step is to evaluate the dynamic pattern mining approach, which promises to be even better suited, due to its streaming and unsupervised character.

Fast and Scalable The processing speed of the current implementation of the statistical classifier is in average 17,385 URIs/second for filtering URIs and in average 3,000 URIs/second for inserts and updates of new obtained information. The performance clearly outperforms the processing speed of the WEKA classifiers. For filtering single URIs our implementation is 319 times faster than the algorithm using the WEKA classifiers and 15 times faster compared to the best achievable processing speed of WEKA (generating an ARFF input file with all URIs $\Rightarrow$ 200 URIs/sec). The performance for training the classifier is in average 27 times faster then the best possible performance of the WEKA implementation (110 URIs/second for the Bayesian classifier) for the same feature combinations.

The evaluation of the focused crawling strategy implemented in the current version of our crawler shows that the crawler continuously gathers the requested documents of content types `image`, `audio` and `video` with the breadth-first crawling strategy. The gradient of the number of fetched relevant documents depends only on the available requested documents in the crawl queue. We can see that the gradient of the curve varies over time and is not constant. Depending on the URIs in the queue, the crawler extracts new links for a fraction of HTML documents that contain more relevant documents as the HTML document in the round before or reverse.

The curve of the näive breadth-first-crawl strategy shows a step function. This clearly shows the filtering and prioritising of URIs leading to relevant documents. In the very beginning of the crawl (<50,000 HTTP lookups) the number of the downloaded relevant documents is zero and then, suddenly, it significantly increases. The explanation for this is that the media-type filter is untrained in the beginning and cannot apply any background knowledge. With more and more visited documents and more obtained information of URIs and their belonging media types, the filter component discovers convenient feature patterns and is capable to identify and filter relevant documents.

The results show that even the a priori method used in combination with a focused crawler can meet the performance requirements of today's Web crawlers. However, performance of Web crawlers can never be good enough, as they usually represent extremely long-running tasks. That is why the stream-based pattern mining approach promises to be a very good choice. We expect it to at least meet the performance we gained in the evaluation presented here, if not even to be capable of producing better results.

6 Related Work

To the best of our knowledge, we are not aware of published work that focuses on the topics of focused crawling for certain media types beside the work of Bachlechner et. al. [Bachlechner *et al.*, 2006], which tries to gather "Web Service Description Language" (WSDL) files on the web. However, the work in hand focuses on the applicability and scalability of different data mining approaches for this task. [Baykan *et al.*, 2009] showed that patterns in URIs can be used for topic classification of the documents identified by the URIs. This is a similar approach to the one presented here. But, the application to media types presents specific requirements and challenges that are not discussed in the field of topic classification.

The list of used classifier algorithms in focused crawlers contains, among others, the application of: a simple naive bayes classifier [Passerini *et al.*, 2001], a k-nearest neighbour clustering algorithm [Ester *et al.*, 2004], a support vector machine [Sizov *et al.*, 2002], a decision tree [Najork and Wiener, 2001] [Li *et al.*, 2005], a neural network [Menczer *et al.*, 2001] and also a solution with hidden markov models [Liu *et al.*, 2004]. They all bear the disadvantage of requiring a supervised approach, similar to the classifier approach presented in this work. None of the works from above considers an unsupervised learning approach, neither they discuss the applicability of stream-mining techniques. Further, none of these works discusses media-type focused crawling.

7 Conclusion

Specialised search engines face major difficulties to discover and gather in a scalable and efficient way structured content of requested media types on the Web. Näive solutions such as gathering URIs via user submissions or crawling the entire Web (existing of 41 billion unique URIs with over 80% of `text/html` documents) for the targeted files do not guarantee a sufficient supply of URIs.

We investigated the problems of specialised search engines in discovering and gathering relevant documents from the Web. We first showed that classification approaches are suited in principle. Afterwards, we proposed an approach for scalable and optimised focused crawling that discovers URIs of targeted media types and extracts meta data in a structured format from the downloaded content. We were able to show that data mining techniques are well suited for implementing fast and scalable focused crawling. However, the choice of applied technique is a rather crucial one. Even if static classifier approaches work well and achieve good performance and accuracy, there is still great potential to increase both. We are eager to integrate and evaluate the pattern mining approach developed for data streams in order to judge on its applicability. We hope that we can increase performance and accuracy by this

even more – while being able to adapt to the usually strict resource limitations that Web crawlers have to face.

References

[Agrawal and Srikant, 1994] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.

[Agrawal *et al.*, 1993] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 207–216. ACM Press, 1993.

[Bachlechner *et al.*, 2006] Daniel Bachlechner, Katharina Siorpaes, Holger Lausen, and Dieter Fensel. Web service discovery - a reality check. In *Proceedings of the 1st Workshop: SemWiki2006 - From Wiki to Semantics, co-located ESWC*, Budva, Montenegro, June 2006.

[Baykan *et al.*, 2009] Eda Baykan, Monika Henzinger, Ludmila Marian, and Ingmar Weber. Purely url-based topic classification. In *18th International World Wide Web Conference*, pages 1109–1109, April 2009.

[Chakrabarti *et al.*, 1999] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: A new approach to topic-specific web resource discovery. *Computer Networks*, 31, 1999.

[Ester *et al.*, 2004] Martin Ester, Hans-Peter Kriegel, and Matthias Schubert. Accurate and efficient crawling for relevant websites. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 396–407. VLDB Endowment, 2004.

[Franke *et al.*, 2006] C. Franke, M. Karnstedt, and K. Sattler. Mining data streams under dynamicly changing resource constraints. In *KDML 2006: Knowledge Discovery, Data Mining, and Machine Learning*, pages 262–269, October 2006.

[Han *et al.*, 2000] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 1–12, New York, NY, USA, 2000. ACM.

[Lausen and Haselwanter, 2007] Holger Lausen and Thomas Haselwanter. Finding web services. In *1st European Semantic Technology Conference*, volume 2007. ESTC, June 2007.

[Li *et al.*, 2005] Jun Li, Kazutaka Furuse, and Kazunori Yamaguchi. Focused crawling by exploiting anchor text using decision tree. In Allan Ellis and Tatsuya Hagino, editors, *WWW (Special interest tracks and posters)*, pages 1190–1191. ACM, 2005.

[Liu *et al.*, 2004] Hongyu Liu, Evangelos Milios, and Jeannette Janssen. Probabilistic models for focused web crawling. In *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*, pages 16–22, New York, NY, USA, 2004. ACM.

[Menczer *et al.*, 2001] Filippo Menczer, Gautam Pant, Padmini Srinivasan, and Miguel E. Ruiz. Evaluating topic-driven web crawlers. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 241–249, New York, NY, USA, 2001. ACM.

[Najork and Wiener, 2001] Marc Najork and Janet L. Wiener. Breadth-first crawling yields high-quality pages. In *Proceedings of the 10th International World Wide Web Conference*, page 114118, Hong Kong, May 2001. Elsevier Science.

[Passerini *et al.*, 2001] Andrea Passerini, Paolo Frasconi, and Giovanni Soda. Evaluation methods for focused crawling. In *Proceedings of the 7th Conference of the Italian Association for Artificial Intelligence, Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[Pearl, 1985] Judea Pearl. Bayesian networks: A model of self-activated memory for evident ial reasoning. In *Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine*, pages 329–334, 1985.

[Quinlan, 1993] Ross J. Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, January 1993.

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

[Sahami *et al.*, 1998] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[Sizov *et al.*, 2002] Sergej Sizov, Stefan Siersdorfer, Martin Theobald, and Gerhard Weikum. Weikum: The bingo! focused crawler: From bookmarks to archetypes. In *Demo Paper, International Conference on Data Engineering (ICDE*, 2002.

[Umbrich *et al.*, 2008] Jürgen Umbrich, Andreas Harth, Aidan Hogan, and Stefan Decker. Four heuristics to guide structured content crawling. In *Proceedings of the Eighth International Conference on Web Engineering*, 2008.

[Umbrich, 2008] Jürgen Umbrich. Discovering and crawling structured content. Master's thesis, University of Karlsruhe (TH), School of Economics and Business Engineering, 2008.

[URL, a] http://citeseer.ist.psu.edu/.

[URL, b] http://scholar.google.com/.

[URL, c] http://seekda.com/.

[URL, d] http://technorati.com/.

[URL, e] http://tools.ietf.org/html/rfc1436.

[URL, f] http://tools.ietf.org/html/rfc959.

[URL, g] http://www.blinkx.com/.

[URL, h] http://www.iana.org/assignments/mime-types/.

[URL, i] http://www.ietf.org/rfc/rfc2396.txt.

[URL, j] http://www.seeqpod.com/.