Web image size prediction for efficient focused image crawling

Article · July 2015

DOI: 10.1109/CBMI.2015.7153609

Symeon Papadopoulos
The Centre for Research and Technology, Hellas
256 PUBLICATIONS **4,720** CITATIONS
SEE PROFILE

Ioannis (Yiannis) Kompatsiaris
The Centre for Research and Technology, Hellas
1,023 PUBLICATIONS **14,035** CITATIONS
SEE PROFILE

Web image size prediction for efficient focused image crawling

Katerina Andreadou, Symeon Papadopoulos, Yiannis Kompatsiaris
Information Technologies Institute (ITI)
Centre for Research and Technology Hellas (CERTH)
Thessaloniki, Greece
Email: {kandreadou,papadop,ikom}@iti.gr

***Abstract*—In the context of using Web image content for analysis and retrieval, it is typically necessary to perform large-scale image crawling. A serious bottleneck in such set-ups pertains to the fetching of image content, since for each web page a large number of HTTP requests need to be issued to download all included image elements. In practice, however, only the relatively big images (e.g., larger than 400 pixels in width and height) are potentially of interest, since most of the smaller ones are irrelevant to the main subject or correspond to decorative elements (e.g., icons, buttons). Given that there is often no dimension information in the HTML `img` tag of images, to filter out small images, an image crawler would still need to issue a GET request and download the respective files before deciding whether to index them. To address this limitation, in this paper, we explore the challenge of predicting the size of images on the Web based only on their URL and information extracted from the surrounding HTML code. We present two different methodologies: The first one is based on a common text classification approach using the n-grams or tokens of the image URLs and the second one relies on the HTML elements surrounding the image. Eventually, we combine these two techniques, and achieve considerable improvement in terms of accuracy, leading to a highly effective filtering component that can significantly improve the speed and efficiency of the image crawler.**

I. INTRODUCTION

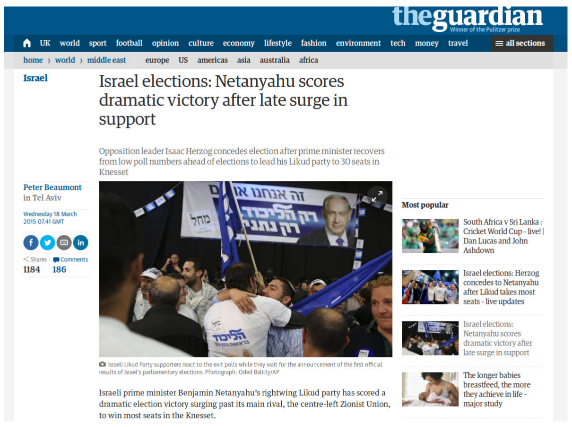
With the widespread use of the Internet as a primary information source by many users and specialists (journalists, analysts, etc.) and with the growing number of people participating and contributing to online social media platforms (Facebook, Twitter, Instagram, etc.) and other kinds of online communities (fora, blogs, etc.), analysing, searching and making sense of the massive amounts of the available Web multimedia content has arisen as a major technical challenge. This development has given rise to intense research analyzing the online context of published content with the goal of evaluating and using it, for instance for collecting supporting material for news reporting and for assessing the truthfulness of a reported story [1].

In a number of image mining and retrieval scenarios, where the analysis and search of large amounts of Web image content is required, it is often necessary to employ focused image crawling, taking as input a number of keywords or topics that interest the user, to collect related images from the Web and social media. Crawling social media networks such as Twitter, Tumblr and Instagram in a targeted way, is often a straightforward task as all of them provide a search API, which supports querying by keyword (though there are also several complications in their use [2], which fall outside the scope of this paper). The main limitation in this case pertains to the scale of collected content: Gathering large quantities of social media items is often impossible due to the fact that most APIs have severe restrictions, specifying rate limits. As a result, Web image crawling is often the only viable option for collecting large amounts of content; yet, much more complicated solutions are necessary in order to deal with the volume and noise of Web content.

978-1-4673-6870-4/15/$31.00 © 2015 IEEE

A number of focused crawling algorithms have been proposed in order to increase the harvest rate (the number of relevant web pages discovered by the crawler) and the target precision (the number of relevant crawl links). Link context algorithms rely on the lexical context of the URL within its parent page [3], graph structure algorithms take advantage of the structure of the Web around a page to find nodes that lead to many relevant Web pages [4], and semantic analysis algorithms utilize ontologies for semantic classification [5]. Yet, no comprehensive focused crawling solution exists in literature that is targeted to image content.

One might use several criteria for evaluating the relevance of a Web image to the supplied keywords: whether the alternate text of the `img` element contains any of the keywords, whether the containing web page title contains any of the keywords, etc. Nonetheless, a common problem is that many images of very small dimensions, which are typically of no interest to the analysis process, may meet such criteria, thus introducing a considerable amount of noise in the collected content.


theguardian
Winner of the Pulitzer prize
UK world sport football opinion culture economy lifestyle fashion environment tech money travel
all sections
home › world › middle east
europe US americas asia australia africa
Israel
Peter Beaumont
in Tel Aviv
Wednesday 18 March 2015 07.41 GMT
Shares 1184
Comments 186
Most popular

Israel elections: Netanyahu scores dramatic victory after late surge in support

Opposition leader Isaac Herzog concedes election after prime minister recovers from low poll numbers ahead of elections to lead his Likud party to 30 seats in Knesset

Israeli Likud Party supporters react to the exit polls while they wait for the announcement of the first official results of Israel's parliamentary elections. Photograph: Oded Balilty/AP

Israeli prime minister Benjamin Netanyahu's rightwing Likud party has scored a dramatic election victory surging past its main rival, the centre-left Zionist Union, to win most seats in the Knesset.

South Africa v Sri Lanka : Cricket World Cup - live! | Dan Lucas and John Ashdown

Israel elections: Herzog concedes to Netanyahu after Likud takes most seats - live updates

Israel elections: Netanyahu scores dramatic victory after late surge in support

The longer babies breastfeed, the more they achieve in life - major study

Fig. 1. Guardian article about Israel elections. Only the main image is relevant to the article topic, while the rest of the images should not be indexed.

For instance, consider a crawl topic around the recent election in Israel. In Figure 1, apart from the main central image which is clearly on topic, there is a multitude of other image elements that are irrelevant: the icons on the left prompting the user to share the article on social networks, the images on the right that link to other articles, as well as banners, logos and icons that belong to the website theme. This fact influences the performance and speed of image crawling solutions, as their main bottleneck pertains to fetching the content. To this end, one might use approaches such as boilerpipe[1], which remove the surplus clutter around the main textual content; however, such approaches are time consuming and in many cases not effective, e.g., when there is no main article and image in the web page.

To this end, this paper proposes a supervised learning solution for automatically predicting the image size solely on the basis of the image URL and the surrounding HTML code. We demonstrate that the proposed image size prediction system is highly accurate and that it can lead to considerable speed-up in the operation of the image crawler.

II. Related Work

Image crawling: Crawling multimedia content is not a new technical challenge. As early as 1999, Sougata et al. [6] developed AMORE (Advanced Multimedia Oriented Retrieval Engine), which is made of two separate components: The Explorer discovers interesting sites based on the user's queries and the Analyzer filters out uninteresting sections from the discovered web sites. In turn, AMORE allows the user to search for relevant images either by providing a text query or an input image. It also uses image URLs to extract useful information, however using a different approach to ours: It automatically assigns keywords to images when crawling based on the web page containing them and one of the places where it looks for keywords is the image URL. Another system for large-scale image retrieval on the Web is Cortina [7]. This collects and indexes images based on visual features and accompanying text and metadata, and prompts the user to enter a text query. Matching images are retrieved and the user has the possibility to choose the ones that best match their search criteria to launch a visual nearest-neighbour search. Finally, as far as video crawling is concerned, LeeDeo [8] is an example of a video search engine that targets academic videos on the Web.

Common Crawl: For the needs of this work, as will become apparent, we used the Common Crawl dataset[2], which is a crawl corpus containing web page data, extracted metadata and text extractions, available from Amazon S3. The Common Crawl is a widely used dataset for many big data analytics, machine learning and web search tasks, thanks to its completeness and accessibility. For instance, the authors in [9] use the Common Crawl and a Wikipedia dump to investigate the frequency distribution of numbers on the Web. They show that, like words, numbers follow a Power law distribution and compare the regularities in the distribution for the two aforementioned paradigms. They find that Wikipedia contains many more symbolic numbers such as model types, whereas the Common Crawl contains many more household numbers such as prices, coordinates, ZIP codes, telephone numbers. Meusel et. al in [10] explore the evolution of the underlying structure of the World Wide Web during the last 10 years by analysing the Common Crawl over a period of time and eventually question whether the heavy-tailed distributions observed in many Web crawls are inherent in the network structure or a side-effect of the crawling process itself. Another use of the Common Crawl dataset is the work by Petrovski et al. [11]. In this work, the authors extracted 1.9 million product offers from 9240 e-shops to analyse the challenges of marking up content with microdata.

URL classification: The experiments conducted in the context of this work are also related to URL classification approaches, which are based on extracting n-grams from the reference text. For instance, in [12], the authors extract n-gram based features from URLs alone and use Support Vector Machines and Maximum Entropy classifiers to classify web pages. In [13], a similar approach is adopted to classify web pages and additionally predict their Pagerank score.

III. Data Collection

For our work, we used a sample of the data from the July 2014 Common Crawl set[3], which is over 266TB in size and contains approximately 3.6 billion web pages.

Since for technical and financial reasons, it was impractical and unnecessary to download the whole dataset, we created a MapReduce job to download and parse the necessary information using Amazon Elastic MapReduce (EMR). The data of interest include all images and videos from all web pages and metadata extracted from the surrounding HTML elements. The output of each MapReduce job resulted in 5GB gzipped JSON files containing documents in the format of Listing 1.

Listing 1. Common Crawl parsed data in JSON format

```
{
   "src":"http://image.architonic.com/
      img_pro2-3/115/9324/eda-m-4-1-01b.
      jpg",
   "alt":"General lighting | Free-standing
       lights | Eda p Floor lamp",
   "w":"",
   "h":"",
   "pageUrl":"http://www.architonic.com/
      pmsht/eda-metalarte_proref/1159325"
      ,
   "domSib":0,
   "domDepth":13,
   "domElem":"img"
}
```

To complete the task, we used 50 Amazon EMR medium instances, resulting in 951GB of data in gzip format. The following statistics were extracted from the corpus:

- 3.6 billion unique images
- 78.5 million unique domains
- ≈8% of the images are big (width and height bigger than 400 pixels)

[1]https://code.google.com/p/boilerpipe/
[2]http://commoncrawl.org
[3]http://blog.commoncrawl.org/2014/08/july-2014-crawl-data-available/

- ≈40% of the images are small (width and height smaller than 200 pixels)
- ≈20% of the images have no dimension information

To have an estimate of the time such an approach could save us, we used the Apache Benchmark[4] to time many random image requests. The average download time for a big image is approximately 300 milliseconds, whereas the average time our classification approach would need is less than 10 milliseconds. So if we attempted to download all images without dimensions from the Common Crawl (720 million images) using 10 download threads (in a single core), it would take approximately 35 weeks. Instead, by only fetching images that were detected to be big, this time would be reduced to less than three weeks. Hence, it is evident that the gain in time by applying the proposed approach would be substantial.

IV. Overview

We propose different supervised learning approaches: The first is solely based on the frequency of the n-grams extracted from the image URLs. Two more variations of this approach are also tested, where the n-grams, which are used as features, are selected based additionally on their correlation with the two classes and not only their frequency. The fourth approach is a variation using tokens instead of n-grams and the fifth one is based on constructing a feature vector from the surrounding HTML element and page (Table IV). Finally, we combine two of the aforementioned approaches into a hybrid approach.

Pre-processing: First, we check whether the image URL, which has been parsed from the Common Crawl dataset is usable and valid. Most of the extracted image URLs are relative and they demand resolving by combining the web page URL and the number of forward slashes in the image URL. For instance, if the web page URL is http://www.example.com/ab/cd/test.html and the image URL is /../img.jpg, the result would be http://www.example.com/ab/img.jpg.

A. Method I: NG

An n-gram in our case is a continuous sequence of n characters from the given image URL. The main hypothesis we make is that URLs which correspond to small and big images differ substantially in wording. For instance, URLs from small images tend to contain words such as *logo*, *avatar*, *small*, *thumb*, *up*, *down*, *pixels*. On the other hand, URLs from big images tend to lack these words and typically contain others. If the assumption is correct, it should be possible for a supervised machine learning method to separate items from the two distinct classes.

To perform the training of the model, we first collected the most frequent n-grams of the training image URLs where $n = \{3, 4, 5\}$. Subsequently we ranked the collected n-grams by frequency. The most frequent n-grams in a set of 2 million URLs are displayed in Table I. Subsequently, for every URL, we constructed the feature vector as follows: for every n-gram of the f most frequent ones, the respective vector position is 1 if the URL contains the n-gram and 0 otherwise. We will refer to this method as NG. We ran our experiments for $f = 1000$, and $f = 2000$. The values for n and f are arbitrary, they are however the result of preliminary testing with smaller datasets.

TABLE I. NG: Most frequent n-grams

N-gram	Frequency	N-gram	Frequency
com	1748886	mag	696830
AAA	1733205	vata	696653
AAAA	1458399	vatar	696536
AAAAA	1185575	avat	696055
age	772729	avata	695924
tar	737723	www	685153
ata	734099	mage	675238
vat	704989	ima	588258
ava	702940	imag	578385
atar	697196	image	577151

B. Method II: NG-trf

The disadvantage of the previous method is that, although the frequencies of the n-grams are taken into account, what is not considered is the correlation of the n-grams to the two classes, BIG and SMALL. For instance, if an n-gram is very frequent in both classes, it makes sense to get rid of it and not consider it as feature. On the other hand, if an n-gram is not very frequent (not one of the first 1000 or 2000 that we take into account in the previous case), but it is very characteristic of a specific class, we should include it in the feature vector. To this end, the feature selection procedure we established is the following:

1) Collect the most frequent n-grams of the training image URLs where $n = \{3, 4, 5\}$ separately for SMALL and BIG images.
2) Rank the two separate sets by frequency.
3) Limit the list sizes by discarding n-grams that are below an arbitrary frequency threshold (in our case less than 50 occurrences in 500K image URLs).
4) For every n-gram of each of the two lists, compute a correlation score, which is the frequency of the n-gram in the first list minus the frequency of the n-gram in the second list.
5) Rank again the two lists by this score. The first members of the two lists and their scores are displayed in Table II.
6) The feature vector is created by choosing equal numbers of n-grams from the two lists. For instance, to extract 1000 features, pick the first 500 n-grams with the highest correlation with the BIG class and the first 500 n-grams with the highest correlation with the SMALL class.

As in the previous case, the feature vector for every image URL is computed by checking if the URL contains the chosen n-grams, and the experiments are conducted again with two different vector sizes, 1000 and 2000. We will refer to this method as NG-trf, standing for term relative frequency.

C. Method III: TOKENS-trf

In a variation of the previous method, we decided to replace n-grams with the tokens produced by splitting the image URLs by all non alphanumeric characters. The employed regular

[4]https://httpd.apache.org/docs/2.2/programs/ab.html

TABLE II. NG-TRF: N-GRAMS AND THEIR SCORE FOR SMALL AND BIG IMAGES

N-gram SMALL	Score	N-gram BIG	Score
humb	33473	600	50775
hum	33266	s1600	44963
thu	31847	s160	44799
thum	31830	s16	44732
thumb	31760	160	42086
ata	27242	201	29156
atar	25143	AAA	22036
vata	25139	mblr	19155
vatar	25128	blr	19154
tar	25011	umblr	19154
ava	24926	umbl	19040
avat	24767	tum	19030
avata	24754	mbl	19023
s72	19157	tumbl	19019
age	16437	tumb	19011
mage	15772	mblr_	16118

TABLE III. TOKENS-TRF: TOKENS AND THEIR SCORE FOR SMALL AND BIG IMAGES

token SMALL	Score	token BIG	Score
png	31510	jpg	65501
gif	22796	s1600	44934
s45	20923	uploads	12504
c	20085	com	9328
s72	19237	photobucket	8960
images	16164	albums	8798
1	8689	s640	8710
thumbnail	5682	content	6805
0	5440	2012	6353
thumb	5171	2013	6331
thumbs	4834	xxxlarge	6186
themes	4815	bp	5733
t	4665	blogspot	5660
large	4314	wp	4191
avatar	4133	2014	3387
small	4005	01	3061
150x150	3536	tumblr	3043

expression in Java is `\\W+` and the feature extraction process is the same as described in steps 1-6 above, but with the produced tokens instead of n-grams. We will refer to this method as `TOKENS-trf`. The tokens with the highest scores for the two classes are displayed in Table III.

D. Method IV: NG-tsrf-idf

A further improvement over the previous method is to increase the impact of the overall frequency of an n-gram on the feature selection. This results in the exclusion of all n-grams with high frequencies in both classes. To this end, we introduce another metric, which is a variation of the tf-idf (term frequency - inverse document frequency). We will refer to it as `tsrf-idf` (term squared relative frequency - inverse document frequency) and it is defined in Equations 1 and 2 for the `BIG` and `SMALL` classes respectively:

$$S_{big}(n) = \frac{f_{big}(n)^2 - f_{small}(n)^2}{f_{big}(n)} \quad (1)$$

TABLE IV. NON-TEXTUAL FEATURES

Name	Description
suffix_JPEG	1 if the image URL has a JPG suffix, 0 otherwise
suffix_PNG	1 if the image URL has a PNG suffix, 0 otherwise
suffix_BMP	1 if the image URL has a BMP suffix, 0 otherwise
suffix_GIF	1 if the image URL has a GIF suffix, 0 otherwise
suffix_TIFF	1 if the image URL has a TIFF suffix, 0 otherwise
domDepth	Depth of the image element in the DOM tree
domSimblings	Number of siblings of the image element in the DOM tree
hasWidth	1 if a width value could be extracted from the image URL, 0 otherwise
width	Extracted width value
hasHeight	1 if a height value could be extracted from the image URL, 0 otherwise
height	Extracted height value
samedomain	1 if the image and originating web page URLs have the same hostname, 0 otherwise
domElement_IMG	1 if the DOM element is <img>, 0 otherwise
domElement_LINK	1 if the DOM element is <link>, 0 otherwise
domElement_A	1 if the DOM element is <a>, 0 otherwise
domElement_EMBED	1 if the DOM element is <embed>, 0 otherwise
domElement_IFRAME	1 if the DOM element is <iframe>, 0 otherwise
domElement_OBJECT	1 if the DOM element is <object>, 0 otherwise
hasAltText	1 if the alt text of the image element is not null, 0 otherwise
altTextLength	Length of the alt text if it exists
hasParentText	1 if the parent element has text, 0 otherwise
parentTextLength	Length of the parent element text if it exists
urlLength	Length of the image URL

$$S_{small}(n) = \frac{f_{small}(n)^2 - f_{big}(n)^2}{f_{small}(n)} \quad (2)$$

The feature extraction process is the same as described in steps 1-6 in subsection IV-B, with the only difference that the computed score is the tsrf-idf instead of just the difference of frequencies, as was the case before.

E. Method V: Non-textual features

An alternative non-textual approach does not rely on the image URL text, but rather on the metadata, that can be extracted from the image HTML element. More specifically, the selected features are presented in Table IV. The idea behind their choice is for them to reveal cues regarding the image dimensions. For instance, the first five features, which correspond to different image suffixes, were selected due to the fact that most real-world photos are in JPG or PNG format, whereas BMP and GIF formats usually point to icons and graphics. Additionally, there is a greater likelihood that a real-world photo has an alternate or parent text than a background graphic or a banner. The features `hasWidth`, `width`, `hasHeight` and `height` of the table are extracted using the regular expression `\\d+x\\d+)+|(w|h|s)_?\\d+|\\d+px|(width|height|w|h)=\\d+|_\\d+\\..`

F. Method VI: Hybrid

Our final approach is a hybrid of the `NG-tsrf-idf` method and method V, which was found to be the best performing combination. `TOKENS-trf` combined with method V was also tested but the results were slightly worse. The goal of the hybrid method is to achieve higher performance by taking into account both textual and non-textual features. Our hypothesis is that the two methods will complement each other when aggregating their results as they rely on different kinds of features: the n-gram classifier might be best at classifying a certain kind of images with specific image URL wording, while the non-textual features classifier might be best at classifying a different kind of images with more informative HTML metadata.

The results aggregation method we use is straightforward: for every image, we construct both feature vectors and use the `NG-tsrf-idf` and the non-textual features classifier. Then for every image in question, we let the two classifiers produce a prediction. The prediction is always a probability pair (P_{small}= probability that the image is small, P_{big}= probability that the image is big). If the classifiers agree, we use this as the final prediction. Otherwise we trust the classifier with the most confident prediction, which means the one with the biggest absolute difference of probability distributions between the two classes. In fact, our decision is guided by Equation 3.

$$|d_{NG}[big] - d_{NG}[small]| + adv > |d_f[big] - d_f[small]| \quad (3)$$

The adv parameter of the previous formula allows us to give a small advantage to the `NG-tsrf-idf` method when deciding, which means that we trust it more, since it delivers better results than the non-textual method. More details on this are provided in section V-B.

V. Evaluation

The aim is to classify the images in the given set into two distinct groups:

`SMALL`	: width and height smaller than 200 pixels
`BIG`	: width and height bigger than 400 pixels

Images with dimensions that lie in between the aforementioned thresholds are ignored.

A. *Methods I-V*

For training we used one million images (500K small and 500K big) and for testing 200 thousand (100K small and 100K big). To avoid complications stemming from the class imbalance problem (one class represented by more instances than the other one), we used equal numbers of small and big images both for training and testing. In fact, we found out that in the dataset, there were approximately ten times more small than big images, which is to be expected, if we take into account all the icons and image resources used for building websites, which do not really represent content but rather decorative elements.

The described supervised learning approaches were implemented based on the Weka library[5]. We performed numerous preliminary experiments with different classifiers (LibSVM, Random Tree, Random Forest), and Random Forest (RF) was found to be the one striking the best trade-off between good performance and acceptable training times. The main paramter of RF is the number of trees. Some typical values for this are 10, 30 and 100, while very few problems would demand more than 300 trees. The rule of thumb is that more trees lead to better performance; however, they simultaneously increase considerably the training time.

[5]http://www.cs.waikato.ac.nz/ml/weka/

TABLE V. Comparative results (F-measure)

Method	RF trees	Features	$F1_{small}$	$F1_{big}$	$F1_{avg}$
NG	10	1000	0.831	0.820	0.825
NG	30	1000	0.840	0.834	0.837
NG	100	1000	0.845	0.840	0.843
NG	10	2000	0.842	0.830	0.836
NG	30	2000	0.853	0.846	0.849
NG	100	2000	0.858	0.852	0.855
NG-trf	10	1000	0.867	0.864	0.866
NG-trf	30	1000	0.871	0.870	0.871
NG-trf	100	1000	0.873	0.872	0.873
NG-trf	10	2000	0.874	0.869	0.872
NG-trf	30	2000	0.880	0.877	0.879
NG-trf	100	2000	0.884	0.882	0.883
TOKENS-trf	10	1000	0.876	0.867	0.871
TOKENS-trf	30	1000	0.887	0.883	0.885
TOKENS-trf	100	1000	0.894	0.891	0.893
TOKENS-trf	10	2000	0.875	0864	0.870
TOKENS-trf	30	2000	0.888	0.828	0.885
TOKENS-trf	100	2000	**0.897**	**0.892**	**0.895**
NG-tsrf-idf	10	1000	0.876	0.872	0.874
NG-tsrf-idf	30	1000	0.883	0.881	0.882
NG-tsrf-idf	100	1000	0.886	0.884	0.885
NG-tsrf-idf	10	2000	0.883	0878	0.881
NG-tsrf-idf	30	2000	0.891	0.888	0.890
NG-tsrf-idf	100	2000	0.894	0.891	0.892
features	10	23	0.848	0.846	0.847
features	30	23	0.852	0.852	0.852
features	100	23	0.853	0.853	0.853
Method	**adv**		$F1_{small}$	$F1_{big}$	$F1_{avg}$
hybrid	-0.05		0.934	0.935	0.934
hybrid	0		0.935	0.935	0.935
hybrid	0.05		**0.935**	**0.935**	**0.935**
hybrid	0.1		0.935	0.935	0.935
hybrid	0.15		0.934	0.934	0.934

The comparative results for different number of trees for the Random Forest algorithm are displayed in Table V. As a measure of accuracy, we use the F1 score (or F-measure), which considers both precision and recall, using a weighted average as illustrated in Equation 4.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (4)$$

The first column of Table V contains the method name, the second one the number of trees used in the RF classifier, the third one the number of features used, and the remaining columns contain the F-measures for the `SMALL` class, the `BIG` class and the average. The reported results lead to several interesting conclusions.

Doubling the number of n-gram features improves the accuracy in all cases. For instance, for 100 trees, the NG method with 1000 n-grams has an average F1 score of 0.843 whereas for 2000 trees, the score increases to 0.855. The improvement is bigger for the `NG` method ($\approx$1.2%), and somewhat smaller for the `NG-trf`, `NG-tsrf-idf` and `TOKENS-trf` methods ($\approx$0.7%).

Adding more trees to the Random Forest classifier improves the accuracy in all cases. For instance, the `NG-trf` method with 1000 n-grams has an average F1 score of 0.866 for 10 trees, 0.871 for 30 trees, and 0.873 for 100 trees. The improvement is greater for all methods when going from 10 to 30 trees than when going from 30 to 100 trees. Additionally, the `NG` method seems to be more influenced by the number of trees in comparison to the other methods.

Overall the `NG-tsrf-idf` and `TOKENS-trf` have the best performance, followed closely by `NG-trf`. Importantly, the `NG-tsrf-idf` and the `NG-trf` methods perform much better than the `NG` method, 5% and 4% respectively, which means that selecting the features (n-grams) instead of just considering the n most frequent ones, really makes a difference in performance. Finally, the performance of the non-textual feature classifier lies in between the simple `NG` and the rest.

The `tsrf-idf` variation was also applied to the `TOKENS` method with the aspiration of improving the `TOKENS-trf` performance, as is the case of `NG-tsrf-idf`. Unfortunately the employed algorithm favours tokens that are highly class-specific (which tend to occur much less often), resulting in many image URLs with empty or almost empty feature vectors. As a result, the classification performance in that case dropped significantly. A legitimate question is why this fact does not influence the `NG-tsrf-idf` method. Our assumption is that class-specific tokens are more rare than class-specific n-grams, mainly because of their size: n-grams have a maximum length of 5 but there is no length limit for tokens. For instance, many of them are longer than 10 characters.

B. Method VI

To evaluate the hybrid method, we use the same test set of 200K images (100K big and 100K small) and we compare the results for different values of adv (Table V). For the `NG-tsrf-idf` classifier, we use the classification model built with 100 trees because of its considerably higher performance and for the non-textual features classifier, the model built with 30 trees, as the one with 100 trees is not significantly better and it requires three times more memory for loading.

One may observe that the hybrid method outperforms all standalone methods, its best result being 4% higher than the `NG-tsrf-idf` best result. Another interesting conclusion is that we get the best result by slightly favouring the `NG-tsrf-idf` method as opposed to the non-textual one ($adv = 0, 0.05, 0.1$), something which was to be expected due to the higher performance of the first.

VI. Conclusion

In this paper, we presented a supervised learning approach for automatically classifying Web images according to their size by only taking into account the image URL and any available HTML metadata. We proposed refinements that improved our text-based method by 4% and then combined the textual and non-textual classifiers to improve the results by an additional 4%. We explained how the data for this experiment was collected and pre-processed and we described the context in which this method can be useful within a Web image crawling system. We made comparisons for different classifiers and showed the effectiveness of the proposed approach by evaluating and comparing the presented methods on images sampled from the Common Crawl dataset. To our knowledge, no other such extensive comparison has been done combining textual features and non-textual features extracted from image URLs and the surrounding HTML code. In the future, we plan to extend our method by applying the n-grams approach to both the alternate and parent text of image elements in order to create two additional classifiers, which we can combine with the existing ones to further increase classification accuracy. Additionally we are interested in testing the proposed approach for the detection of more fine-grained image characteristics in addition to the size; for instance, to distinguish between landscape and portrait images or photographs and graphics.

Acknowledgment

This work was supported by REVEAL project, partially funded by the European Commission under contract no. FP7-610928.

References

[1] C. Boididou, S. Papadopoulos, Y. Kompatsiaris, S. Schifferes, and N. Newman, "Challenges of computational verification in social multimedia," in *Proceedings of the Companion Publication of the 23rd Inter. Conference on World Wide Web*, 2014, WWW '14, pp. 743–748.

[2] S. Papadopoulos and Y. Kompatsiaris, "Social multimedia crawling for mining and search," *IEEE Computer*, vol. 47, no. 5, pp. 84–87, 2014.

[3] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur, "The shark-search algorithm. an application: Tailored web site mapping," in *Proceedings of the Seventh Inter. Conference on World Wide Web 7*. 1998, pp. 317–326, Elsevier Science Publishers B. V.

[4] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused crawling: A new approach to topic-specific web resource discovery," in *Proceedings of the Eighth International Conference on World Wide Web*, NY, USA, 1999, WWW '99, pp. 1623–1640, Elsevier North-Holland, Inc.

[5] A. Maedche, M. Ehrig, S. Handschuh, R. Volz, and L. Stojanovic, "Ontology-focused crawling of documents and relational metadata," in *Proceedings of the 11th International World Wide Web Conference, WWW 2002, Honolulu, Hawaii*, May 2002.

[6] S. Mukherjea, K. Hirata, and Y. Hara, "Amore: A world wide web image retrieval engine.," *World Wide Web Journal*, vol. 2, no. 3, pp. 115–132, 1999.

[7] T. Quack, U. Mönich, L. Thiele, and B. S. Manjunath, "Cortina: A system for large-scale, content-based web image retrieval," in *Proceedings of the 12th Annual ACM International Conference on Multimedia*. 2004, MULTIMEDIA '04, pp. 508–511, ACM.

[8] Dongwon L., Hung-sik K., Eun Kyung K., Su Y., Johnny C., and Jeongkyu L., "Leedeo: Web-crawled academic video search engine," in *Tenth IEEE International Symposium on Multimedia (ISM2008), December 15-17, 2008, Berkeley, California, USA*, 2008, pp. 497–502.

[9] W. R. van Hage, T. Ploeger, and J. Hoeksema, "Number frequency on the web," in *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, 2014, WWW Companion '14, pp. 571–572.

[10] R. Meusel, S. Vigna, O. Lehmberg, and C. Bizer, "Graph structure in the web — revisited: A trick of the heavy tail," in *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web*, 2014, WWW Companion '14, pp. 427–432.

[11] P. Petrovski, V. Bryl, and C. Bizer, "Integrating product data from websites offering microdata markup," in *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web*, 2014, WWW Companion '14, pp. 1299–1304.

[12] R. Rajalakshmi and C. Aravindan, "Web page classification using n-gram based URL features," in *Fifth International Conference on Advanced Computing (ICoAC)*. 2013, pp. 15–21, IEEE.

[13] M.-Y. Kan and H. O. N. Thi, "Fast webpage classification using URL features," in *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, New York, NY, USA, 2005, CIKM '05, pp. 325–326, ACM.