Quantifying and Analysing Advertisements in Web Pages for Spam and Low-Quality Content Detection

JASON BOTHA, University of KwaZulu-Natal, South Africa

The World Wide Web has billions of pages of content. Many of these pages feature digital advertisements, mostly to enable a site to maximise potential revenue from displaying such content. A portion of these websites display excessive and intrusive advertisements to the point of negatively affecting the experience of the reader. Such pages also typically offer virtually no worthwhile content and serve as spam pages. It is not uncommon for these websites to attempt to lure readers to a page primarily to expose them to advertisements.

By formulating an objective rating scale on web pages based on the advertisements they have, it can reveal whether a page may be showing what could be considered an excessive number of advertisements. A browser extension was created that interacted with web pages in order to scan their contents and identify advertising elements. Taking into consideration various factors such as the advertisement areas, paragraph interruption frequency, and the vertical distribution type of advertisements and paragraphs, a scanning script was able to deterministically rank compatible web pages and suggest a level of quality.

CCS Concepts: • **Information systems → Spam detection**; **Page and site ranking**; **Content ranking**.

Additional Key Words and Phrases: advert analysis, page quality, spam detection, client-side analysis

1 INTRODUCTION

As the World Wide Web has developed, it has become a primary means of accessing and sharing information, with an estimated 4.72 billion active internet users [6] as of April 2021. Being such a ubiquitous part of life for many has resulted in it being a vector of opportunity for both good and malicious acts. Tim Berners-Lee, credited with inventing the World Wide Web, noted in a 2019 press release to celebrate the web's 30th anniversary:

> And while the web has created opportunity, given marginalised groups a voice, and made our daily lives easier, it has also created opportunity for scammers, given a voice to those who spread hatred, and made all kinds of crime easier to commit. [1]

Berners-Lee further lists three sources of dysfunction that he believes are issues in the modern web. Source two is listed as:

> System design that creates perverse incentives where user value is sacrificed, such as ad-based revenue models that commercially reward clickbait and the viral spread of misinformation. [1]

Advertising is an unavoidable part of the web experience, with more than US$157 billion spent on web advertising alone in 2020 [3]. This way of reaching an audience can be beneficial for genuine uses, but can also be abused. This is because anyone can set up a website and get paid to display advertisements. The more views these adverts get, the more revenue that can be generated and paid to the site owners. As outlined by Markines et al. [11], this financial reward is the primary motivation for many social spam cases. All that is needed is to receive credit is to display adverts and attract sufficient viewers to pages that have said adverts. This naturally leads advertising abusers to i) put as many adverts as possible on a web page, with little consideration for the actual content or user experience, and ii) promote these pages in different ways to attract viewers. This is part of the "perverse incentive" referred to by Berners-Lee in [1]. It is possible that this issue of adverts being overused and contributing to a poor web experience could be alleviated to some extent. This could be done by scanning web pages in an automated manner to determine the probability of that page displaying too many advertisements, and subsequently creating an undesirable browsing experience.

Author's address: Jason Botha, University of KwaZulu-Natal, South Africa.

2 LITERATURE REVIEW

Methods to determine the likelihood of a web page having many adverts revolve primarily around analysing the word content for spam markers as described in [13, 15, 17]. Phrases such as 'free', 'trial', and '% off' are used as identifiers and the static word content in the whole page is analysed for such words. The frequency of such identifiers is then compared to the whole data set and a confidence rating can be reached on whether the said page is spam or not. The same approach is used for categorizing emails as possible spam. This is effective in cases where the web page is delivering spam as text-based content. This however cannot be utilized for a page where the text is good quality content, as the spam may only be in the form of dynamic or image adverts. In such a case the page would bypass the filter as dynamic content and images are not read by the text analyser. It could be possible to use this existing text analysis alongside a new technique to enhance results.

Web pages are already scanned in some way to determine their quality. Google Search may penalize a website listing in search results if they detect that it has a significantly poor user experience [10]. The page loading speed and 'Core Web Vitals', consisting of metrics like first input delay and visual stability, are used to judge the user experience. If a page has many advertisements, the loading time will generally be longer. This approach is an efficient method as it simply measures the time to load and does not need to process the source code. The context here is that a longer loading time is detrimental to the user experience regardless of it being caused by advertisements loading, or if it was other resources loading. Such a method cannot be used for measuring advertisement frequency specifically.

Papers by Parmar et al. [19] and Shukla et al. [8] describe methods for detecting advertisements in web pages for the purpose of removing or hiding these as part of advertisement blocking. These can be found in software such as AdBlock Plus. These methods do not result in formal feedback that may suggest the number of adverts blocked and quality of the web page, in order to rate pages. Consequently, a 'remove and hide' approach to web adverts cannot be used to detect low-quality pages. They do however share the need to detect and

identify advertisements, and the same identification methods can be adapted to fulfil part of the proposed page rating. Something that can be used to our advantage in detecting advertisements is the fact that many web adverts use a common sizing schema as outlined in [21]. This means that there are standards that most adverts will adhere to in order to qualify for the majority of advertisements slots on web pages. Any elements found on a web page that have a known standard advert size can already be assumed to belong to an advert, taking into consideration any other attributes of an advert element. This is however not an absolute guideline, and dynamic adverts also exist which can adapt to other sizes also described in [21]. Attributes such as the element size as discussed as well as the name and source of the item would need to be considered. Alcic and Conrad [5] outline this approach by looking at the semantic and geometric representation of elements in the DOM (Document Object Model), as well as the depth and distance of elements in the DOM tree. It is concluded that the semantic approach was most accurate, although also most computationally expensive. Their context was looking at the web page as a whole, but still applies to identifying adverts. Their clustering methods are considered further on.

Ad-blocking software like the aforementioned AdBlock Plus typically intercepts the resources that are being loaded on a page. If the origin of a resource matches a filter [2], the resource would be blocked from loading and subsequently hidden on the page. Such a filter would include known advert vendors and keywords that have been found to load advertising resources. This is efficient at identifying adverts as it does not require the page source code to be analysed. It does require a pre-defined list to populate the filter, and as such may suffer from a so-called 'cold start' where a dataset needs to be in place before the program comes into effect. Further, this can only determine the frequency of advertisements on a page, and not the area taken up nor the invasiveness of them. Facebook has a similar approach to this to some extent as seen in [18]. They analyse the traffic and source of posts using a service called Click-Gap:

> Click-Gap looks for domains with a disproportionate number of outbound Facebook clicks compared to their place in the web graph. This can be a sign that the domain is succeeding on News Feed in a way that doesn't reflect the authority they've built outside it and is producing low-quality content. [18]

This could address the issue to some extent on Facebook's site specifically. The disadvantage is that it needs previous data to use Click-Gap and also has the cold-start issue, and such an approach being implemented globally is expensive in terms of overhead and storage.

User experience on a web page is another factor to consider, as it needs to be established exactly how web adverts can cause degraded experience. The user experience on a web page is determined by several factors, including simplicity, interface functionality, content, and credibility [7]. If the content is not relevant or is low quality, it already negatively impacts the impression of the reader, and they will be more inclined to leave the page. The same applies to the layout and simplicity of the page design and contents – a poor design or intrusive content will subconsciously disinvest attention in the page. The perception of how good or bad a design is a subjective issue. However, there are some guidelines, as described in [20] for example, that can provide a more empirical approach to determining the user experience. This involves answering different sections of analysis with yes or no. This approach could provide an approximated evaluation of the user experience, but it would be very complex to programmatically implement this analysis and automate it with human accuracy. Human intervention is always needed for such analysis unless an advanced script to simulate human behaviour is developed.

With regards to how adverts specifically can negatively affect the user experience, Ying et al. [14] and Zhulidin [22] found that the size and area of advertisements is the primary factor that page viewers consider when judging the level of advert intrusion, along with consistent or many pop-up style adverts. A large-sized interstitial advert was found to be more intrusive and distracting than several smaller adverts. The trend noted was that if the primary content was not obstructed nor interfered with, the impact and associated annoyance of adverts was less. In the context of our topic, however, a single interstitial advert showing instead of multiple smaller ones does not necessarily imply that the page is already worse off in quality – there would have to be persistent and alternate advertisements showing as well for the page to be considered as exploiting advert content, as a single interstitial advert by itself would not make a page a low-quality, ad-based page. Summarising the previous two paragraphs, it appears that the most logical way to determine the way that adverts affect user experience is to use the metrics described by Ying et al. [14] and Zhulidin [22]. The properties of advertisements like the size, location, and total area are metrics that are objectively measurable. The higher these advertisement metrics are, the more they adversely affect the user experience as they detract from the content, functionality, and credibility of a page.

A trivial manner in which to classify pages as having too many adverts is to use a linear ranking scale. As the sum of the metrics reaches certain thresholds, the probability of the page being poor quality increases. However, there are alternate methods of classification that could be better, particularly with clustering techniques. An example of this approach is shown in [16], where a violent crime analysis is done on the United States. Each state has four metrics associated with it – the population and three crime statistics. This is related to how our advert scan can return several metrics for each advert element on the page. Fuzzy C-Means clustering is used in this context as described as by Premasundari and Yamini in [16]. We could define a set of clusters to cover a pre-defined range of page qualities and draw conclusions as to how individual adverts may contribute to affecting the page. It is then possible to use their collective outcomes to decide for the whole page. This particular approach taken in [16] was effective, and although not in the context of web pages, it shares a similar dataset format.

The clustering mentioned earlier by Alcic and Conrad [5] used partitioning, hierarchical, and density-based methods. It is concluded that the density-based clustering was most effective in classification, although in our context we should still explore other possible methods.

3 METHODS AND TECHNIQUES

3.1 Software Design and Modelling

Modern web pages are mostly dynamic and capable of adjusting the content to the viewport size, as well as loading content via background HTTP requests after the main page request has already terminated. This results in a client-side page scan being inherently more feasible than a server-side analysis on incoming HTML code in the context of our analysis. A web browser extension or page with iFrame embedding are two of the possible client-side approaches. This approach is outlined in Fig. 1.

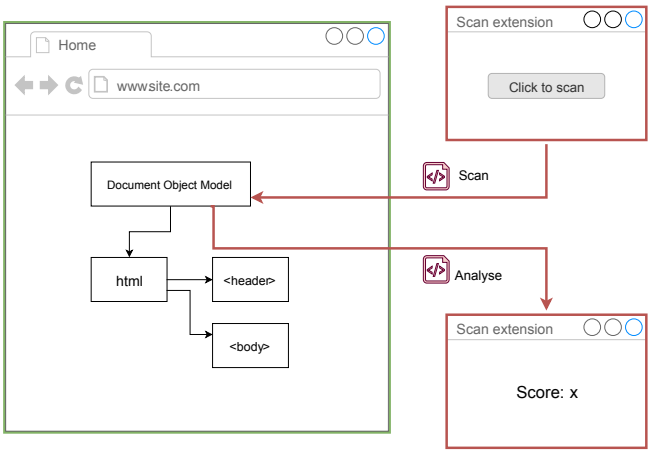

Fig. 1. The architecture of the analyser extension and interaction with the web page to scan

The analyser is a separate environment from the web page being displayed, but has full permissions to access it. The same approach can be taken for the iFrame embedding, except the analyser code is contained in the parent page and not an extension. Specifically, a Chrome extension was used as this is the author's default browser. The software structure of this extension is detailed in Fig. 2. JavaScript files are used to facilitate the scanning of the current active web page. For the iFrame, the scan code can be directly inserted into the parent page.

The scan.js file provides several functions in order to achieve the functionality of interacting with the web page and scanning the contents of it by accessing its Document Object Model (DOM). jQuery is a JavaScript library that greatly abstracts the necessary code for different browsers and version to manipulate the DOM. Some of the functionality is asynchronous in nature and needs to be explicitly defined to return a JavaScript 'promise' to retain the desired sequential order that functions are called in. The higher level overview of what these functions need to fulfil is shown in appendix 11.

3.2 Development and Implementation

The standardised HTML (Hypertext Markup Language) in a given page that makes up the Document Object Model is the primary entry of analysis. The elements within this DOM each contain numerous attributes with associated information that can be extracted to give more context on the element and page as a whole. An example of such an element and its structure is shown in Fig. 3.

The attributes and their values in the DOM elements are most relevant to the scope of our analysis. A list of regular expressions can be used to check whether the attributes and associated values of an element match known advert element types. An example of such an expression is `[name*='advert_block']`. If an element's name attribute were to contain the string 'advert_block', a reference to that DOM element is added to the return list. Nearly 22 000 expressions were adapted from ad-blocking software [4] and greatly modified to work with the jQuery [9] library. jQuery is capable of accepting a regular expression as a parameter to return all HTML elements that match that regular expression. This functionality in particular is useful as implementing this feature without such a library would be tedious.

Once the library returns the list of positive identifications, it is necessary to filter this list and remove some elements for three reasons:

(1) Advert elements can be nested in multiple parent elements, of which more than one may be returned in the list of adverts if they have positive attributes. The calculated area of the advert will be greater than it actually is.
(2) Some advert elements may not be configured correctly or unable to fully load. These items do not typically display to the end user, but still exist in the DOM as small-sized elements.
(3) Icons and small image elements that form part of a page's core structure may be picked up as advert elements, resulting in false positives.

Resultingly, we can remove an element from the list if:

(1) The element overlaps with another element, with a majority of area intersecting.
(2) The height or width of the element is sufficiently small e.g., less than 50px.

At this stage, it is assumed that a reference to all the advert elements has been returned and the elements have been filtered for likely imperfections. The same process is then applied for paragraph elements by using the "<p>" tag, with the exception that small (under 50px in length or bread) elements are not removed.

Before proceeding with the analysis of the page contents through the identified advert and paragraph elements, one consideration to make is how the width of a screen affects the layout of a website's content. In cases where the screen resolution is greater than what the site was designed for, the content is often simply padded by margins. In higher resolution monitors, this padding can approach >60% of the page area. Analysing this page without trimming this excess margin could grossly misrepresent the area that the reader interprets as the actual viewport. An alternative approach could be to still count the padding margin as it can arguably still contribute to the page experience, or consider it to have a lower weight. For the purpose of being able to reproduce a scan on different desktop screens, such margins will be trimmed to fully discount the padding area from the total page area.

In order to trim this padding margin area and analyse the content distribution, a virtual grid that overlays the web page is used.

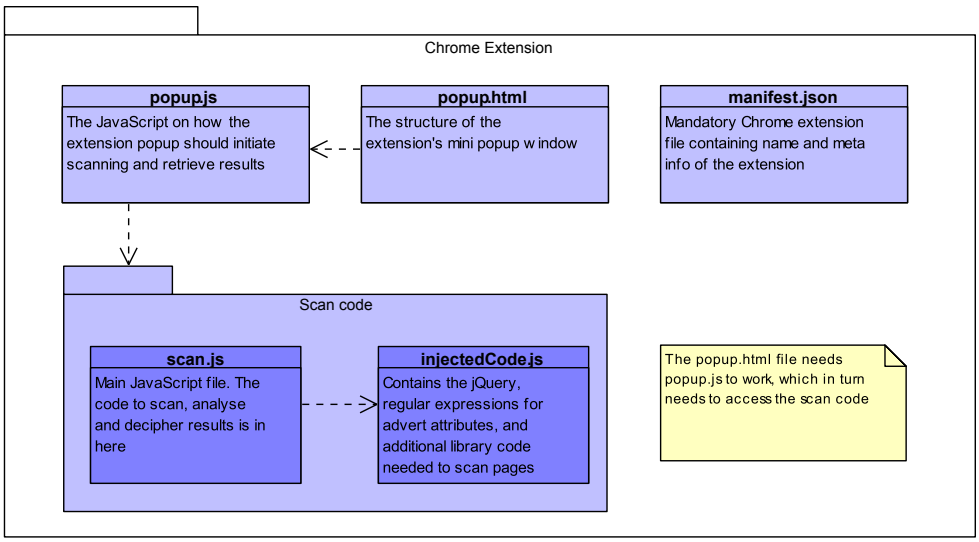

Fig. 2. The software structure of the extension and the code it interacts with to scan

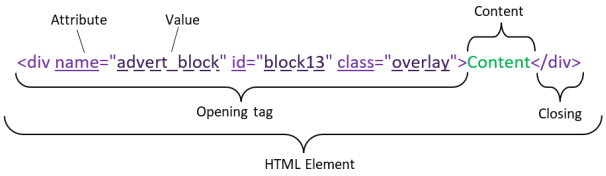

Fig. 3. Components of an HTML element

The purpose of this grid is to represent where elements are on the web page, but without using the original pixel values as a unit of measurement. This way, it becomes significantly more efficient to calculate overlapping elements and determine the overall distribution of elements; the length of the grid object could easily reach several million otherwise, with viewport widths and lengths usually exceeding 1000 pixels. To achieve this, the four offset distances (left, right, top, bottom) of an element are converted from pixel offset values to grid block offsets with `gridBlockOffset = round(pixelOffset/gridBlockWidth)` where `'gridBlockWidth'` is the pixel width of a single grid block, and `'pixelOffset'` is a pixel offset.

This two-dimensional virtual grid can then be simplified into its horizontal and vertical components with two one-dimensional lists, one of which will represent the distribution of elements along the y-axis, and the other one the x-axis. These distribution provide insight into how the content is concentrated and distributed. In the example grid in Fig. 4, the grid has a width of 26, and maintains a height as large as needed. To trim the padding columns, we detect that columns 1 to 4, and 23 to 26 are edge columns as they do not have any positive matching advert or paragraph elements in them. These columns remain intact in our grid, but it is noted that the total page area is now `(26-8)/26 = 69%` as large as before to remove padding. Note that only the advert elements are highlighted in this example.

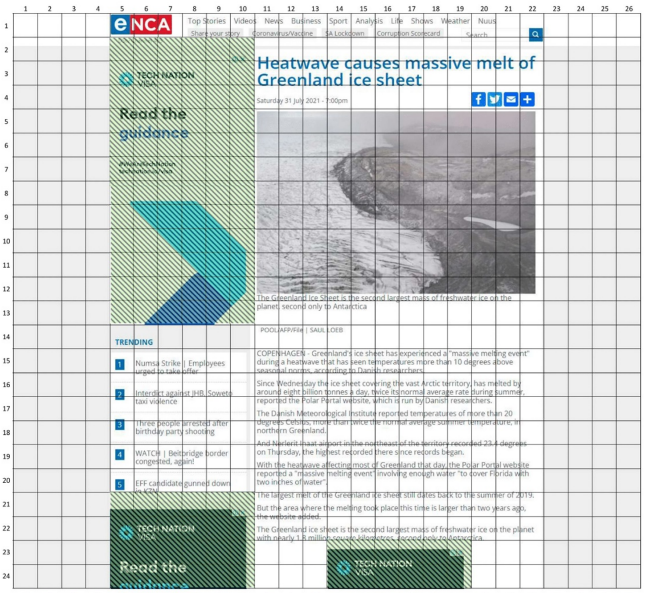

Fig. 4. A virtual grid example, with advert-positive blocks highlighted

Fig. 5 shows how the advert contents are distributed along the x-axis. A graph can also be generated for the y-axis distribution. This same grid and distribution analysis is done on paragraph elements. The type of content distribution and density on the page can then be inferred from these graphs, although the horizontal graph is not used further as it does not provide relevant information beyond trimming margins.

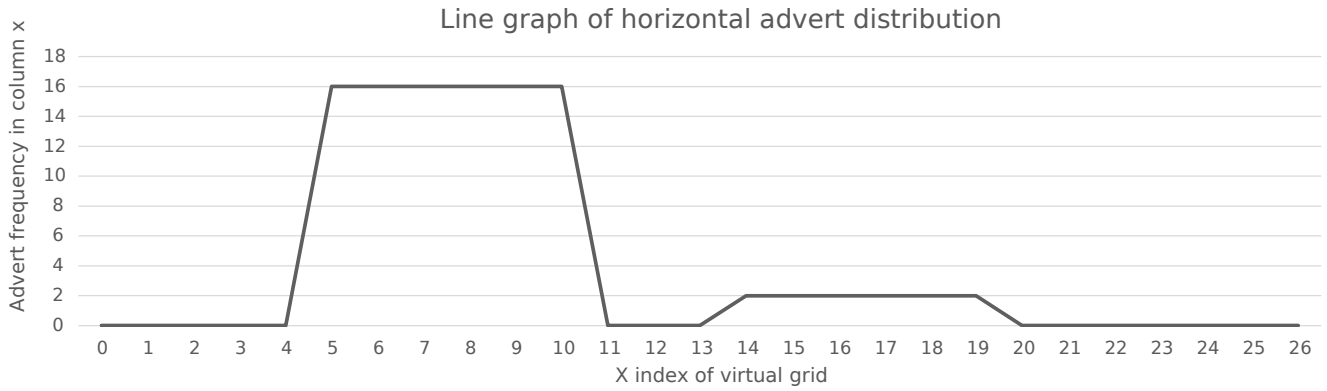

Fig. 5. The horizontal advert distribution of the virtual grid seen in Fig. 4

If the accumulative frequency of the vertical elements is compared to a perfectly linear vertical accumulation (i.e., the contents are equally distributed), the difference in these frequencies will say whether the distribution of these elements is concentrated in the top, middle, or bottom of the page. This difference between the actual and linear distribution can be processed into a simplified representation. The distribution of a middle-dense page and the simplification can be seen below in Table 1.

Decile	1	2	3	4	5	6	7	8	9	10
Linear Distibution	329	658	987	1316	1645	1974	2303	2632	2961	3291
Actual Distribution	132	195	290	600	1400	2250	2800	3050	3150	3291
Difference	197	463	697	716	245	-276	-497	-418	-189	0
Simplified Representation	1	1	1	1	1	-1	-1	-1	-1	-

Table 1. Derivation of the simplified distribution representation

In this example, the value of 3921 in the 10th decile is the number of grid blocks that contain advertising elements. The class row goes to 1 if the difference is positive, -1 if negative, and 0 in cases where the difference is sufficiently small. The sum all of the class values is assigned to a value Σ, and the number of times the classes switch values going from the first to last decile is assigned to α. In this table, Σ = 1 and α=1.

The Σ value indicates to which extent the actual distribution was above or below the equal distribution line. The α value tells us how often the actual distribution graph crossed the equal distribution line. The “High” and “Low” values can be defined as approximately:

Metric	Outcome
Σ	High if magnitude > gridWidth/4, else Low
α	High if > 1, else Low

Table 2. Assigning a ’High’ or ’Low’ class to Σ and α

The significance of the Σ and α values is outlined in Fig. 6:

This approach can accommodate most of the results and provide an approximated conclusion of how the elements are vertically distributed.

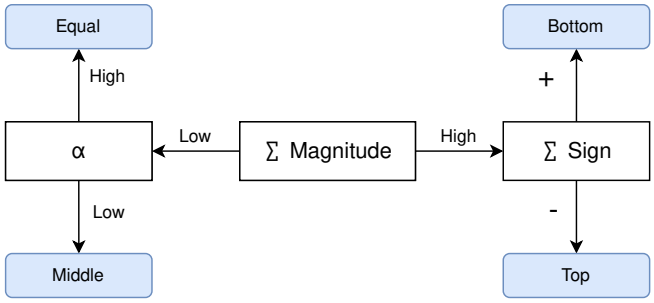

Fig. 6. Decision tree to determine the vertical element distribution given Σ and α

By comparing the vertical distribution types of adverts and paragraphs, the number of times that an advert interrupts a paragraph can be deduced, as well as if the page has a higher density of adverts in the top, middle, or bottom of the page. This is needed as in the event a web page has a bottom-heavy advert distribution and middle/top-heavy paragraph distribution, the effects of the advert elements will be intuitively less as the reader focuses on the text at the top part of a page. The inverse is also true. Fig. 7 illustrates some of the possible vertical distribution patterns and their line graphs. Paragraph interruptions by adverts can be found on a more granular analysis: when the advert graph increases, the paragraph graph simultaneously decreases.

To conclude the scanning, the script returns the following values to the extension environment:

(1) Number of adverts.
(2) Proportion of page area that adverts take up, with margin trimming.
(3) Number of times that a paragraph is interrupted by an advert.
(4) Distribution type of advert and paragraph elements.
(5) A formal score based on the above numbers:
    (a) A number from 0 to 9, where 0 is assigned for 0% advert coverage and 9 for >60%.

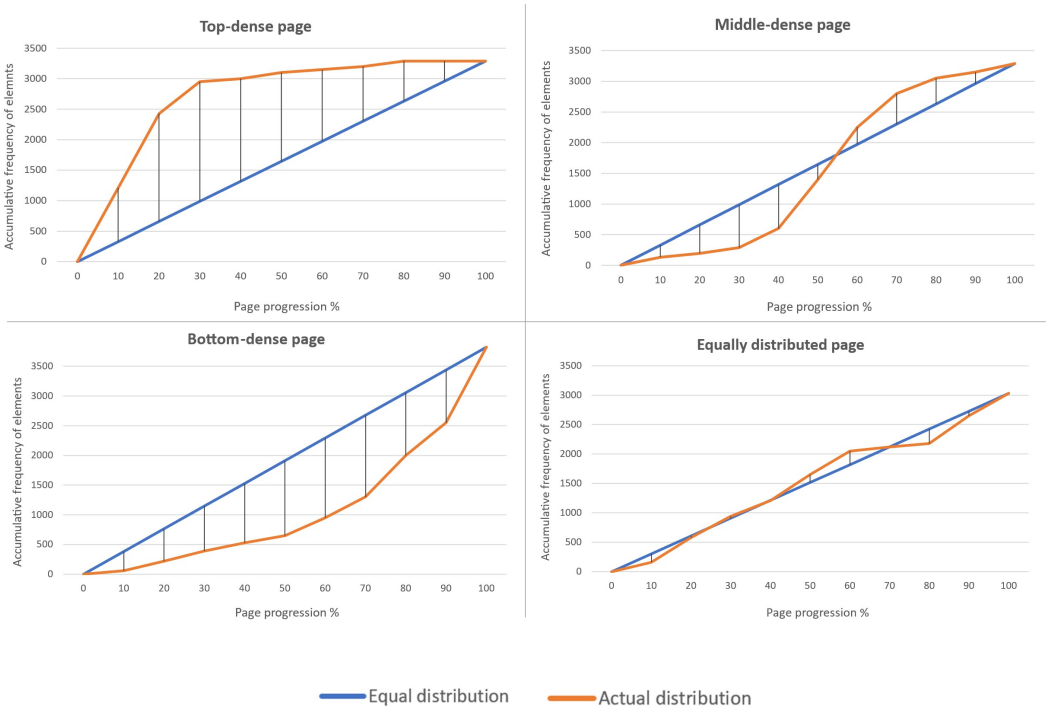

Fig. 7. Some possible types of vertical distributions in web pages

(b) A letter depending on the type of content distribution:
  (i) "A" for approximately equal, or unknown, distribution.
  (ii) "B" for when the majority of advert elements are at the bottom of the page and the paragraph content precedes this.
  (iii) "C" for when there is a significant number of advert interruptions in paragraphs.

The "B" type distribution would favour the page in these sense that the adverts are not part of the primary focus area, while the "C" type implies that the page is worse as the adverts are more integrated with the focus area. A page with a score of 5C is likely subjectively worse for most people than one with a score of 8B.

4 RESULTS AND DISCUSSION

4.1 Environment

The initial approach of using an HTML iFrame element to host the web pages of interest, and then interacting with that iFrame with JavaScript in the parent frame, was not successful. This is as a result of many websites implementing the "same origin" policy and explicitly blocking embedding as a modern security practice. Attempts at using a proxy server were not successful in spoofing content sources. Further, when the iFrame web page did connect and load, the embedded adverts and other external content were still mostly affected by the same origin policy.

A browser extension then seemed to be the most appropriate approach, given how extensions are able to directly interact with and modify web pages. For the context of our application, this actually seems the most logical implementation that can be taken, despite more complexity being involved.

4.2 Advert Element Detection

Two means of detecting adverts were explored.

(1) The size of elements were compared to known common advert sizes. If an element matched a known size within a 5px margin of error, it was presumed to be an advert. This was highly inaccurate; the results are displayed in Fig. 8a. The modern web is aligned to dynamically generate and insert content as well as work on many different screen sizes, so using fixed advertising sizes like in the past would now be a technical liability.
(2) The attributes of each element were checked against a list of regular expressions. The confusion matrix can be seen in Fig. 8b.

Attribute filtering was significantly more successful in identifying advertisements. Interestingly, the adapted expression list was primarily made of the "class" attribute at 60%, with the "id" attribute being the remaining items. It was noted that advertising hosts sometimes use seemingly random domains, meaning that any "src" expression could be quickly outdated. This could lend itself to why the regular expression list did not use the "src" attribute. However, several "src" expressions were manually added to help identify common false negatives.

		Predicted	
		Not advert	Is advert
Actual	Not advert	0	53
	Is advert	28	362

(a) Confusion matrix using attribute detection.

		Predicted	
		Not advert	Is advert
Actual	Not advert	0	36
	Is advert	356	27

(b) Confusion matrix using size detection.

Fig. 8. Confusion matrixes for different identification techniques

4.3 Element Filtering

Some hierarchical advert elements were still duplicated despite provisions made to remove these overlaying duplicates. This may lead to the scan returning a higher frequency of advert elements than there really are on a given page. However, the effect of this was largely mitigated, as several adverts were often grouped into one parent element in some sections.

4.4 Element Processing

The width (number of horizontal partitions) of the virtual grid plays an important role in how close to reality the returned advert area is. A width too small would not provide sufficient granularity and overstate the true area, but a width too big would begin to slow the scan down. Interestingly, the optimal width range that was found was unexpectedly small. When 3 constant pages were scanned with varying width sizes, it was apparent that widths greater than 9 partitions did not noticeably improve results. The relationship between the grid width and calculated advert area is shown in Fig. 9.

Out of 22 sample pages, 2 pages were not able to be automatically scrolled by the script. This is either from the site making provisions to stop automated tools, or most likely, there was some incompatibility with the structure of their page. In such pages, the script cannot be used as the results returned are very inaccurate – there were negative and >100% advert areas returned. This shows how important it is to scroll the page wholly first and load all elements that are set to lazy load.

4.5 Determining Distribution Types

Two approaches were tested when determining the vertical distribution type of elements, given a Σ and α value. These two values are calculated from the simplified distribution representation of a page. Such a representation is described in Table 1, with the list of [1, 1, 1, 1, 1, -1, -1, -1, -1]. It is assumed that conflicting Σ and α values (e.g high sigma magnitude, high alpha) are treated as a neutral 'Equal' distribution. While the example in Table 2 splits the distribution into 10 parts to show the analysis, 20 were used in implementation for better granularity.

A decision tree as in Fig. 6 can approximate the type of distribution based on the sign and magnitude of Σ and α. This method proved simple and successfully classified 6 real-life testing examples. It is sensitive to how the "High" and "Low" magnitudes are defined, so these need to be adjusted accordingly.

A JavaScript machine learning library [12] implementing a neural network (NN) was also experimented with, using the Σ and α values as inputs and the anticipated distribution type as the output. Using 22 training examples, the classifier was 100% accurate on 6 real-life testing examples. The issue with this approach is that while a model can be loaded instead of training the NN each time it is run, Chrome extensions are not allowed to directly access local files. This forces us to either ensure a server can serve the trained model in a file on demand, or train the model each time which results in dramatic (>4 second) time delays and creates issues with keeping the JavaScript functions asynchronous downstream. Subsequently, it is not used further.

4.6 Scan Results

In cases where the pages had a higher advert area, the advert elements were often densest at the bottom of the page, after the reading content. Out of 8 web pages in the dataset that had advert areas in excess of 20%, 4 were found to have middle & bottom, or bottom, distribution types. The reading content in these pages was not perceived by the author as notably affected by advert elements.

Conversely, if the advert area is high and there are several advert interruptions in paragraphs, the effect on user experience can be perceived as much more significant. It is necessary to rank such pages more harshly.

To introduce another level of ranking independent of advert area, a letter was appended to the final score. This indicated whether:

(1) the advert distribution was insignificant, equal, or unknown (neutral effect),
(2) the page intentionally discarded the user experience by interrupting paragraphs several times (negative effect), or
(3) it prioritized the reading content and added adverts last (positive effect).

An example score is shown in Fig. 10 For each scan on the 22 page dataset, the result was compared to the page contents, and in most cases, it performed well and returned a conclusion close to what the author approximated. In some pages, the distribution type returned was not expected and some fine-tuning was done on the "Low" and "High" parameters to fix this.

5 CONCLUSIONS AND FUTURE WORK

5.1 Conclusion

This paper explores a different approach in identifying low quality and spam web pages, and attempts to establish the feasibility and best methods in client-side analysis. Two notable conclusions can be reached after methodical development and testing.

Firstly, existing methods of web page quality analysis are done only through word analysis. There are no known methods that consider other elements rendered on the client side. This sets a difficult precedent with no formal ranking method existing for a topic (page quality) that is inherently subjective for each person. By using objective attributes of a page such as the advert area, paragraph interruptions, and distribution type, an approximate and

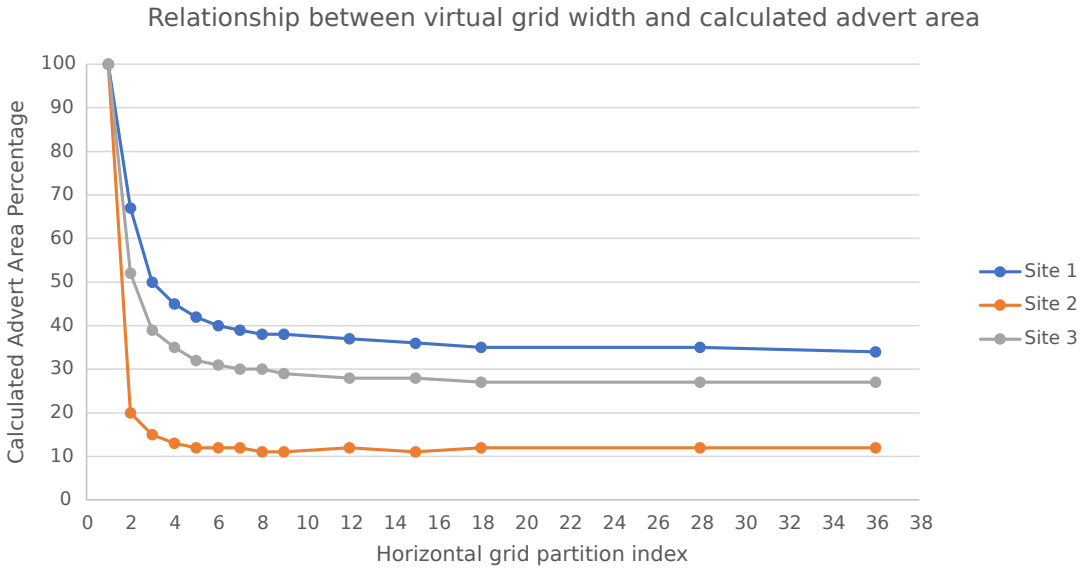

Fig. 9

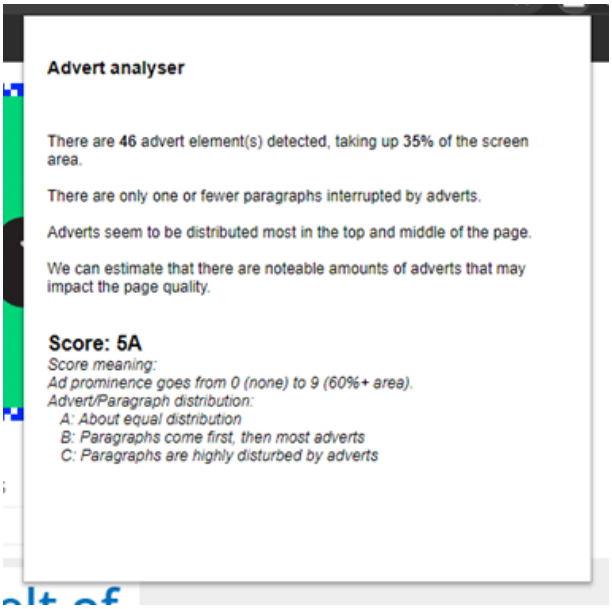

Fig. 10. Result breakdown from a testing scan

predictable score can be assigned to a page that indicates the quality of the contents.

Secondly, it is very difficult to achieve a solution that is compatible with all web pages and environments. Some web pages could not be scrolled by the script, which made the analysis attempt mostly futile. There also exist many different ways of integrating advert elements in a page, with countless possible attributes and values making advert detection fall short of being fully accurate. Further, the result given by the script could be affected if the screen size, network quality, or browser interpretation of the source code are not under ideal conditions. The open nature of the web means that only so much can be standardized.

5.2 Future Work

It should be investigated whether a server-side analysis of static HTML in a web page can achieve a similar result in quality and spam detection. If a page can be analysed server-side, independent of JavaScript and 3rd party content, it would lend itself to larger scale usage as such an approach would be significantly more efficient. The caveat of many 3rd party resources being excluded from the main HTTP response, as they load asynchronously after the main page has loaded, could prove difficult to overcome. Using the client-side analysis first to deduce a rating on a page as a benchmark could prove beneficial.

Another point of interest from a psychological view is how the content of an advert impacts the perceived impact on page quality. It is reasonable to think that an advert with a white background and a few words of black text would have a substantially lower impact on the user experience than an advert that has flashing colours and animations – in such cases, should the analyser take this into account, and to what extent? This same logic leads one to wonder if an advert on the edge of the screen has less impact than one in the middle. Having a formal way of deducing the impact of a single advert element could help the analyser be more accurate.

REFERENCES

[1] 2018. About common sizes for responsive display ads. https://support.google.com/google-ads/answer/7031480

[2] 2020. Timing for bringing page experience to Google Search. https://developers.google.com/search/blog/2020/11/timing-for-page-experience

[3] 2021. Digital Advertising - United States | Statista Market Forecast. https://www.statista.com/outlook/dmo/digital-advertising/united-states

[4] 2021. EasyList Open Source Filter Lists. https://easylist.to/easylist/easylist.txt

[5] Sadet Alcic and Stefan Conrad. 2011. Page segmentation by web content clustering. *Proceedings of the International Conference on Web Intelligence, Mining and Semantics - WIMS '11* (2011). https://doi.org/10.1145/1988688.1988717

[6] Tim Berners-Lee. 2018. 30 years on, what's next #ForTheWeb? https://webfoundation.org/2019/03/web-birthday-30/

[7] Z Colman. 2021. What is User Experience and Why is It Important For Users And Websites? https://creatitive.com/what-is-user-experience/

[8] Xin Deng, Lunqing Hou, and Fei Wang. 2019. Web advertisement detection using Naive Bayes. *Journal of Physics: Conference Series* 1187 (04 2019), 042023. https://doi.org/10.1088/1742-6596/1187/4/042023

[9] JS Foundation. 2019. jQuery. https://jquery.com/

[10] Simon Kemp. 2021. Digital 2021 April Statshot Report. https://datareportal.com/reports/digital-2021-april-global-statshot

[11] Benjamin Markines, Ciro Cattuto, and Filippo Menczer. 2009. Social spam detection. *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web - AIRWeb '09* (2009). https://doi.org/10.1145/1531914.1531924

[12] ml5js. 2021. ml5js/ml5-library: Friendly machine learning for the web. https://github.com/ml5js/ml5-library

[13] Shafi'i Muhammad Abdulhamid, Maryam Shuaib, Oluwafemi Osho, Idris Ismaila, and John K. Alhassan. 2018. Comparative Analysis of Classification Algorithms for Email Spam Detection. *International Journal of Computer Network and Information Security* 10 (01 2018), 60–67. https://doi.org/10.5815/ijcnis.2018.01.07

[14] Alexandros Ntoulas, Marc Najork, Mark Manasse, and Dennis Fetterly. 2006. Detecting spam web pages through content analysis. *Proceedings of the 15th international conference on World Wide Web - WWW '06* (2006). https://doi.org/10.1145/1135777.1135794

[15] Wladimir Palant. 2006. Adblock Plus and (a little) more: Investigating filter matching algorithms. https://adblockplus.org/blog/investigating-filter-matching-algorithms

[16] M Premasundari and C Yamini. 2019. A Violent Crime Analysis Using Fuzzy C-Means Clustering Approach. *Online ICTACT Journal on Soft Computing* (2019), 3. https://doi.org/10.21917/ijsc.2019.0270

[17] Hetal R. Parmar and Jayant Gadge. 2011. Removal of Image Advertisement from Web Page. *International Journal of Computer Applications* 27 (08 2011), 1–5. https://doi.org/10.5120/3316-4555

[18] Guy Rosen. 2019. Remove, Reduce, Inform: New Steps to Manage Problematic Content. https://about.fb.com/news/2019/04/remove-reduce-inform-new-steps/

[19] Kankana Shukla and Ben Choi. 2014. Web Page Advertisement Classification. *International Journal of Computer and Electrical Engineering* 6 (2014), 54–58. https://doi.org/10.7763/ijcee.2014.v6.793

[20] Kaisa Väänänen-Vainio-Mattila and Minna Wäljas. 2009. Developing an expert evaluation method for user eXperience of cross-platform web services. *Proceedings of the 13th International MindTrek Conference: Everyday Life in the Ubiquitous Era on - MindTrek '09* (2009). https://doi.org/10.1145/1621841.1621871

[21] Lou Ying, Tor Korneliussen, and Kjell Grønhaug. 2009. The effect of ad value, ad placement and ad execution on the perceived intrusiveness of web advertisements. *International Journal of Advertising* 28 (01 2009), 623–638. https://doi.org/10.2501/s0265048709200795

[22] Andrey Zhulidin. 2019. How to implement advertising monetization and not to kill UX of your product. https://uxdesign.cc/how-to-implement-advertising-monetization-and-not-to-kill-ux-of-your-product-cd759f23868e

APPENDIX

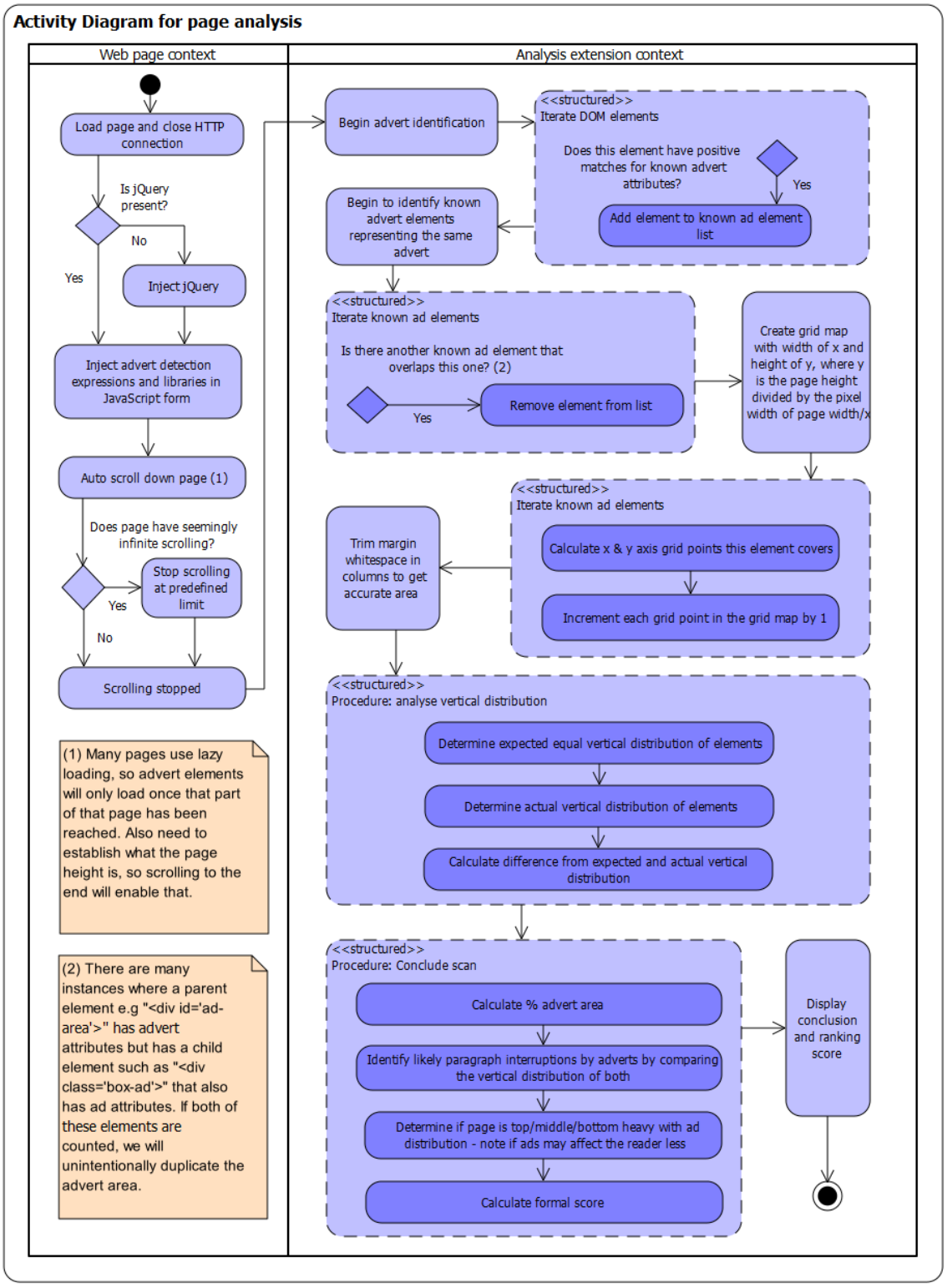

Fig. 11. The flow of functions in analysis