Studying Google WebLight Transformations using deep learning techniques

Muhammad Adil Inam[*]
20100180@lums.edu.pk
Lahore University of Management Sciences
Lahore, Punjab

Ammar Tahir[*]
20100212@lums.edu.pk
Lahore University of Management Sciences
Lahore, Punjab

Ihsan Ayub Qazi
ihsan.qazi@lums.edu.pk
Lahore University of Management Sciences
Lahore, Punjab

Zartash Afzal Uzmi
zartash@lums.edu.pk
Lahore University of Management Sciences
Lahore, Punjab

Zafar Ayub Qazi
zafar.qazi@lums.edu.pk
Lahore University of Management Sciences
Lahore, Punjab

ABSTRACT

Google WebLight is a free transcoding service launched by Google that provides minified web pages on the fly to serve slow clients by performing multiple transformations and compressions, such that pages load faster while saving data. In our work, we perform a detailed empirical analysis on a dataset of around 2000 original and web light transformed pages and evaluate some of the claims made by Google. We also learn the web light transformations and optimizations using machine learning techniques such as pattern matching and an encoder-decoder model with a promising accuracy of 92%. These transformations prove helpful in understanding the impact of Web Light service on page quality and the QoE and can help further reduce the web complexities that are associated with slower connections and low-end devices.

KEYWORDS

Minification, Google WebLight, Deep Learning

ACM Reference Format:
Muhammad Adil Inam, Ammar Tahir, Ihsan Ayub Qazi, Zartash Afzal Uzmi, and Zafar Ayub Qazi. 2018. Studying Google WebLight Transformations using deep learning techniques. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Improving user experience on the internet has always been the primary motivation that has led to the improvement and advancement of a multitude of internet protocols and architectures. In the modern world, there are over 3.6 billion unique users on the internet and a large portion of this traffic belongs to population of developing nations. Hence, it is extremely critical to ensure that the internet users from developing countries do not encounter huge page load times due to the limitations extended by the dearth of resources.

To address this problem, Google launched its Web Light service in 2017. Web Light services to cater faster and lighter pages to the users who access the web on slow mobile clients.âĂŕIts utilities can be summarized to the provision of transcoded web pages on the fly to serve slow clients by performing multiple transformations and compressions, such that pages load faster while saving data. This consequently improves the browsing experience of over a million users who struggle with slow internet speed. According to the analysis pledged by Google, Web Light ameliorated pages load four times faster than the original page and utilized 80 percent fewer bytes in totality. It further claims that the decrease in PLT results in a 50% increase in traffic to these pages [5]. While there is no shortage of bluster, there is a paucity of independent, empirical analysis to evaluate the claims given above. Moreover, although a significant number of previous studies have attempted to understand the webpage complexities [2][6] and suggest transformations to reduce the PLTs [3][4][7], no such work has been carried yet to appreciate and understand the different transformations that are undertook by the Web Light service.

It is essential to learn these transformations due to a veracity of reasons, two of which this paper will consider for the ambit of the purpose it aims to serve. Firstly, an in-depth analysis of these transformations can help further reduce the complexities that are associated with slower connections and low-end devices. This has the ability in enhancing the user experience. Secondly, these transformations can also prove helpful in understanding the impact of Web Light service on page quality and the QoE.

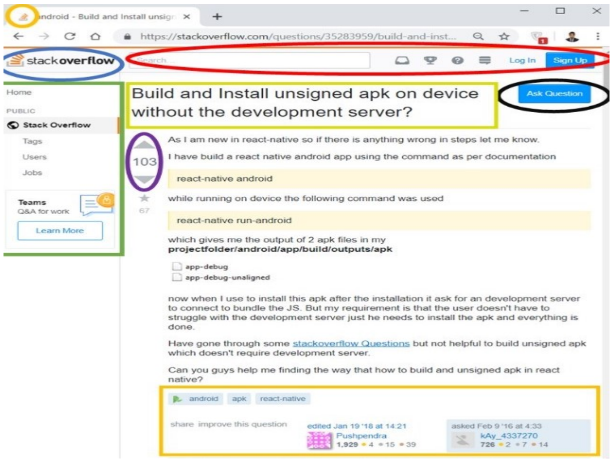
In our work, we perform (i) a detailed empirical analysis on a dataset of 2000 original and web light transformed pages and evaluate some of the claims made by google and (ii) learn the web light transformations and optimizations using machine learning techniques such as pattern matching and encoder-decoder machine translator models.

After collecting the dataset of around 2000 original and transformed pages, we perform an external analysis of the objects fetched

[*] Both authors contributed equally to this research.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456



Build and Install unsigned apk on device without the development server?

As I am new in react-native so if there is anything wrong in steps let me know.

I have build a react native android app using the command as per documentation

```
react-native android
```

while running on device the following command was used

```
react-native run-android
```

which gives me the output of 2 apk files in my **projectfolder/android/app/build/outputs/apk**

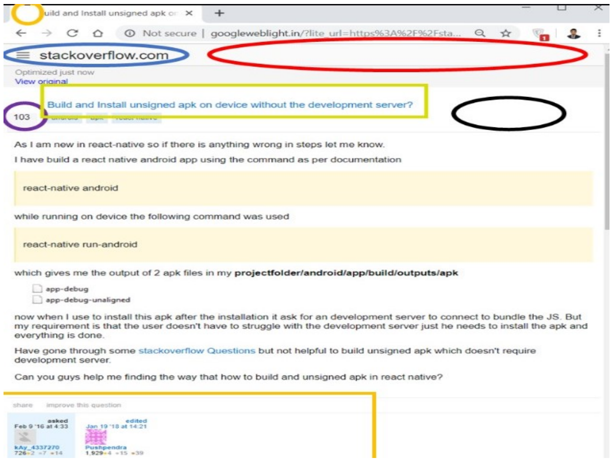
- app-debug
- app-debug-unaligned

now when I use to install this apk after the installation it ask for an development server to connect to bundle the JS. But my requirement is that the user doesn't have to struggle with the development server just he needs to install the apk and everything is done.

Have gone through some stackoverflow Questions but not helpful to build unsigned apk which doesn't require development server.

Can you guys help me finding the way that how to build and unsigned apk in react native?

Figure 1: A normal web page accessed through a desktop browser.



Build and Install unsigned apk on device without the development server?

As I am new in react-native so if there is anything wrong in steps let me know.

I have build a react native android app using the command as per documentation

```
react-native android
```

while running on device the following command was used

```
react-native run-android
```

which gives me the output of 2 apk files in my **projectfolder/android/app/build/outputs/apk**

- app-debug
- app-debug-unaligned

now when I use to install this apk after the installation it ask for an development server to connect to bundle the JS. But my requirement is that the user doesn't have to struggle with the development server just he needs to install the apk and everything is done.

Have gone through some stackoverflow Questions but not helpful to build unsigned apk which doesn't require development server.

Can you guys help me finding the way that how to build and unsigned apk in react native?

Figure 2: A Google WebLight transcoded web page (Same page as last figure).

by a webpage and internal analysis of the underlying html. Our results display a 3 to 4 times reduction in the transformed page size on average, validating the above-mentioned claim made by Google.

Moreover, the empirical analysis of the dataset provides us with certain interesting insights: (i) around 70% of the objects fetched are html in transformed pages as compared to 10-15% in original pages. (ii) The transformed pages make almost no request to any of the non-origin servers. (iii) The transformed pages fetch no external CSS and JS objects. (iv) WebLight embeds a number of its own scripts in the base HTML page and (v) The count of distinct tags in transformed pages reduce from 150 to 25, validating the transformations of different html tags into a single one.

After an extensive pragmatic analysis, we develop a pattern matching technique of our own. We first identify a hierarchy of tags (e.g. html, head, body, div, h1) against page text elements since text remains almost constant in both the transformed and original pages. Then, using string matching techniques, we identify how these tag hierarchies transform in the transcoded page. We then develop NMT Encode-Decode model to learn these transformations with a promising validation accuracy of around 92% on a dataset of around 14000 tag sequences obtained from 1284 web pages. Finally, we developed a technique to get a webpage back given a set of tag sequences thus outputting an actual HTML page at the end. We achieved this by building m-ary trees from tag sequences and then merging these m-ary trees into one tree.

2 RELATED WORK

There have been a lot of work in this area, from understanding complexities of web to realizing mobile devices as bottleneck in developing countries. Some selected papers are as follow:

Web Complexity: This paper identify a set of metrics to characterize the complexity of websites both at a content-level (e.g., number and size of images) and service-level (e.g., number of servers/origins).It presents a detailed analysis of 1700 websites and studies complexities associated to loaded objects, number of servers, content requiring utilities like flash etc. [2] Our empirical analysis follows a similar pattern to this work and includes both a content level and service level analysis of the collected dataset.

HTTP Compression: This paper discusses prevalent techniques for web page compression. It presents a novel elastic compression framework that automatically sets the compression level to reach a desired working point considering the instantaneous load on the web server and the content properties. It also performs a detailed study of the top 500 websites with respect to their compression properties and current practices. HTTP compression can help solve issues where network is the bottleneck instead of low-end devices. The main motivation behind this work is to eventually decrease PLTs by decreasing the page sizes using efficient compression techniques. [7]

Mobile devices in Developing world: If network is not the bottleneck, mobile devices used to access web may have very small processing powers. This paper does in-depth analysis of mobile device characteristics from Pakistan using a dataset of âĹij0.5 million subscribers from one of the largest cellular operators in Pakistan. It also identifies device-level bottlenecks in connectivity to internet. [1]. This paper identifies the possible scenarios where the device may act as a bottleneck. This suggests that a transcoding service such as web light should consider the device implications as well as the network conditions before performing certain desired transformations.

3 DATA COLLECTION

To understand and learn the patterns and transformations, a considerate dataset of original and corresponding transcoded pages is required. However, the large-scale automated data collection presented a few challenges: (i) Few pages were not transformed or optimized by the web light service due to technical and privacy concerns, (ii) Redirects to the original page after the page renders, and (iii) censorship of some domains in Pakistan. For overcoming the first challenge, we used regexes to identify non-transcoded pages. For overcoming the second challenge, we used the get method that downloads the html page directly without rendering it in a browser.

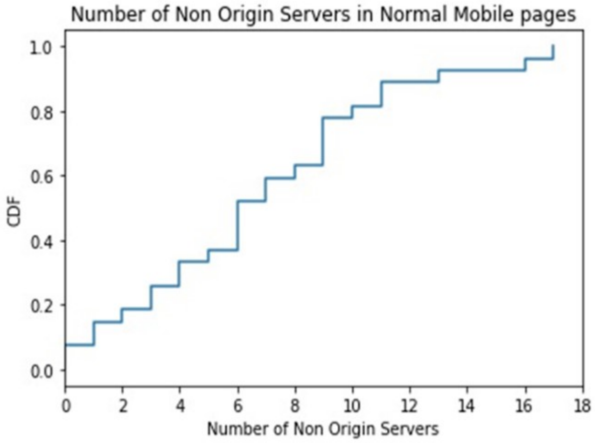


Figure 3: CDF plot of Non-origin servers in Normal mobile pages.

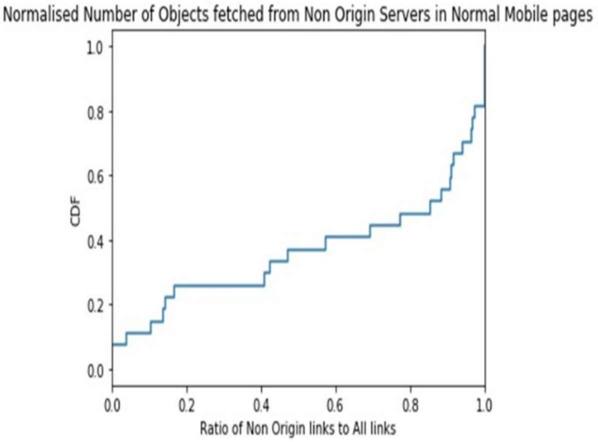


Figure 4: CDF of normalised number of objects fetched from non-origin servers in Normal pages

For overcoming the third challenge, we used regexes as well as performed automated data cleaning.

As the Web Light service is specifically designed for low-end mobile devices with slow internet speeds, the goal was to collect the dataset for such a device. Therefore, the wget method was used with the specific headers of a low- end mobile device: Samsung Galaxy Young. We used AlexaâĂŹs top ranked pages and collected a dataset of around 2000 original and WebLight transcoded pages. Moreover, a dataset of around 1000 non-transcoded pages was also collected for future analysis.

For each webpage, we collected a total of four different versions as stated below.

(1) Low end mobile webpage
(2) Low end mobile webpage transformed via Web Light
(3) Desktop version of web page
(4) Desktop version of web page transformed via Web Light

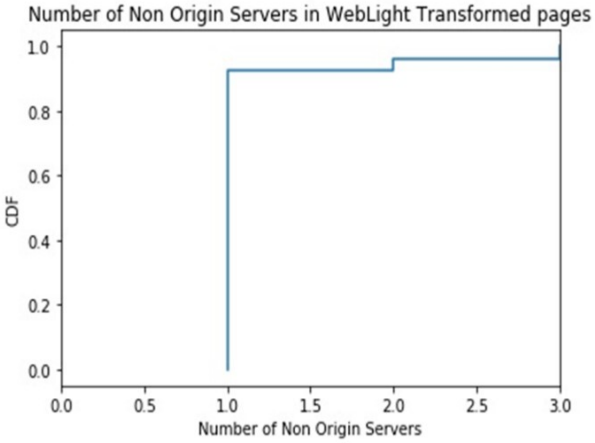


Figure 5: CDF plot of Non-Origin servers in Transcoded page.

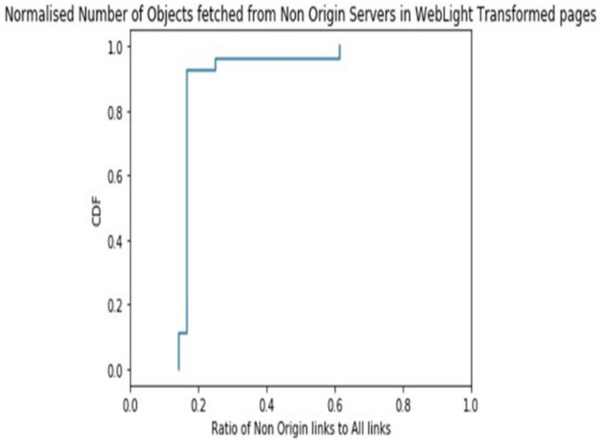


Figure 6: CDF of normalised number of objects fetched from non-origin servers in Transcoded pages

One of the interesting observations was that the Web Light server responded with exactly the same HTML page in case of both a desktop and a mobile request. This suggested that Web Light responds with a standard web page regardless of the host machine making the request. Additionally, it was noted that resolving a domain name on a slow network took a considerate amount of time for downloading the original page. Whereas, WebLight performed the domain resolving part on its own, thus reducing overall load time of the webpage. The exact change in PLTs was not recorded since wgetâĂŹs speed results are not very accurate and present certain other limitations.

Our dataset consists of only the landing pages of top Alexa domains. In the future, we aim to repeat this analysis for non-landing pages as well.

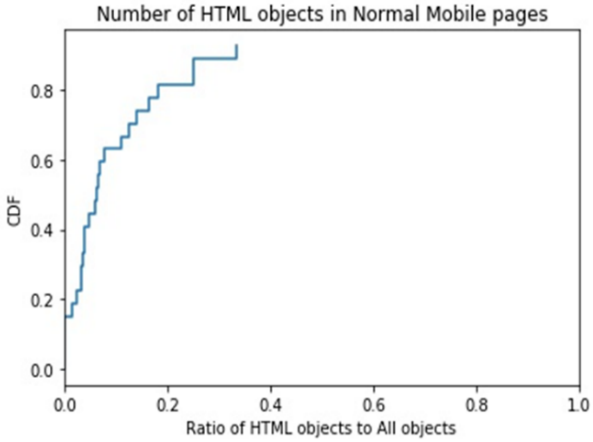


Figure 7: CDF plot of normalised number of HTML objects in normal Mobile pages.

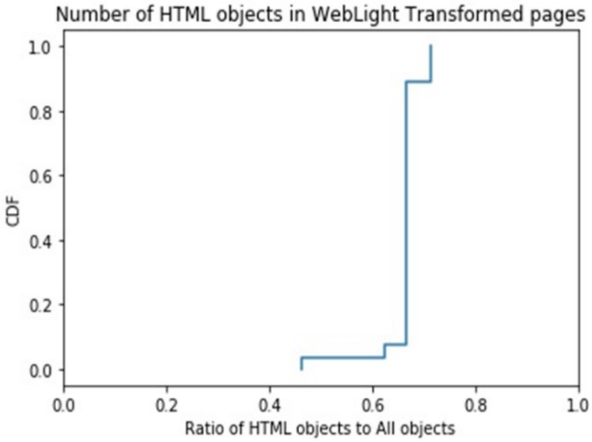


Figure 8: CDF plot of normalised number of HTML objects in Transcoded pages

4 DATA ANALYSIS

We performed an extensive empirical data analysis to familiarize ourselves with the dataset and better understand the underlying problem and challenges. The data analysis part can be further categorized into visual analysis, the external analysis of the objects fetched and the internal analysis of the underlying HTML.

4.1 Visual Analysis

As a first step towards understanding the basic visual transformations, we analyzed the behavior of Google Web Light on certain web domains. We randomly sampled a number of pages, both static and dynamic, from the collected dataset and visually analyzed the page transformations. Some of the interesting visual observations include: (i) changes in page-formatting such as fonts, text size, text and image placements etc. (ii) decreased page content such as removal of ads, snack bars, top banners, navigation bars, search bars, menu bar, images etc. (iii) image compression (iv) decreased page interactivity and removal of dynamic content. Figure 1 and 2 show a detailed visualization of these transformations.

4.2 External Analysis of the Objects Fetched

A webpage makes a number of requests on average to fetch external JS, CSS, Image objects etc. These requests can be further categorized into origin and non-origin. Origin requests are made to origin servers while non-origin requests are made to non-origin servers. In the following sections, we discuss the breakdown of origin and non-origin request and servers as well as the percentage ratio of different objects fetched by a webpage. This analysis was performed on a subset of our dataset by downloading the respective HAR files of each webpage.

4.2.1 ***Non-Origin content.*** **Server Comparison:** The results very prominently display that the number of non-origin servers are greater in number for original pages as compared to transformed pages. Figure 3 and 5 show a side-by-side CDF comparison of non-origin servers in both the original and transformed webpages. This behavior is expected as almost all the page content is served via the google WebLight servers in the transcoded pages. However, there are some requests to non-origin servers present in those pages as well. These servers include domains usually owned by google such as google analytics, gstatic etc. It was interesting to see that google was in some cases embedding analytics scripts without prior consent from the content providers

Requests Comparison: It can be clearly observed from figure 4 and 6 that around 70-80 % of the total requests are made to non-origin servers in case of original pages, where this significantly decreases to around 20% in case of transformed pages. This can be explained due to a decrease in non-origin servers in case of transformed pages.

4.2.2 ***Percentage ratio comparison of Objects Fetched.*** Following are details of objects fetched:

HTML Objects: The ratio of html objects to total objects fetched is significantly higher in transformed pages as seen in Figure 7 and 8. This can be explained by the fact that transformed pages usually make no external CSS or JS requests. Therefore, HTML objects make around 65 to 70% of the total objects fetched on average in case of transformed pages.

JSS and CSS Objects: The transformed pages in the sample dataset made no external CSS or JS object requests. It was observed that all the CSS styles and necessary JS scripts are embedded in the base html page in the transformed pages.

Image Objects: The ratio of external image objects to total objects fetched is significantly lower in transformed pages as most of the images are completely embedded inside the html page in the transformed pages. In the average case, the ratio decreases from 50% to 20% from original to transformed pages as seen in Figure 9 and 11.

4.3 Internal Analysis of the Underlying HTML

This analysis was extensively performed on a subset of 1284 pages in our dataset. We parsed all the html pages, both original and transformed, and performed an extensive tag level analysis. Moreover,

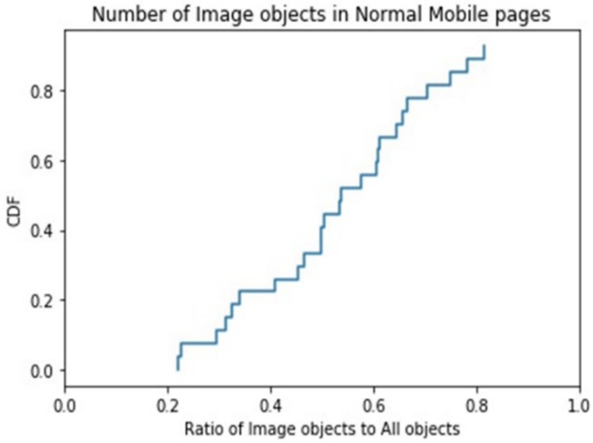


Figure 9: CDF plot of normalised number of image objects in normal Mobile pages

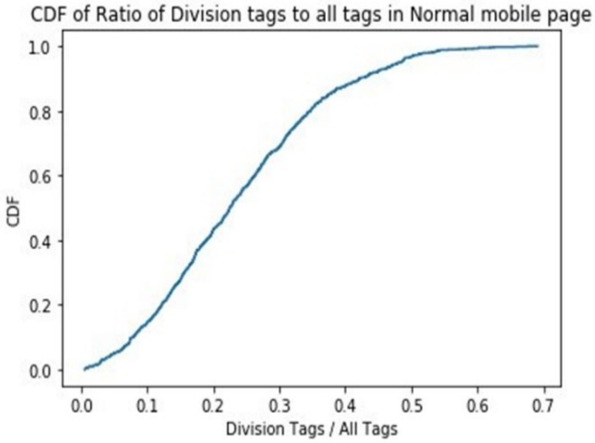


Figure 10: CDF plot of normalised number of division tags in normal Mobile pages

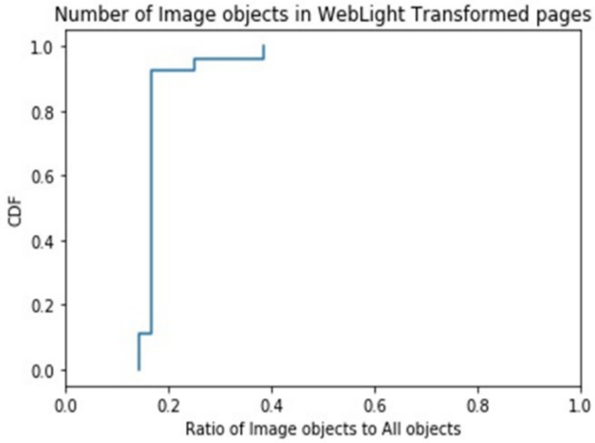


Figure 11: CDF plot of normalised number of image objects in transcoded pages

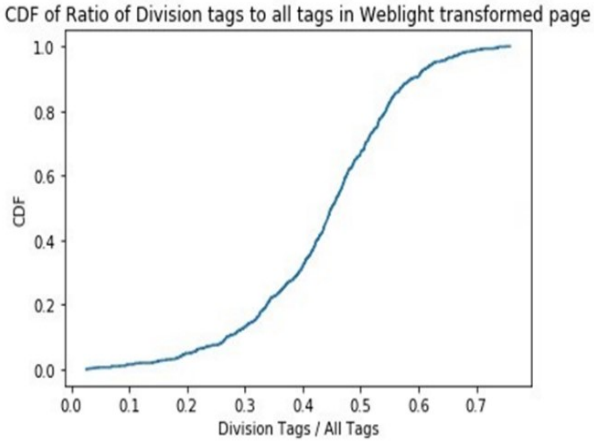


Figure 12: CDF plot of normalised number of division tags in transcoded pages

the sizes of the original and transformed HTML pages were also compared.

4.3.1 ***Page Size Comparison****.* The plot in Figure 13 shows the decrease in page sizes for transformed pages. In the average case, the plot shows that there is a 3 to 4 times decrease in page size for transformed pages in the average case. These results validate the claims made by google regarding page size reduction.

4.3.2 ***Tag Level Analysis****.* We parsed the html pages to identify the unique tags in both the original and transformed pages. The analysis provided us with several interesting insights: (i)number of distinct tags significantly decreasing from 150 in original pages to 25 in transformed pages. (ii) absence of some tags in transformed pages e.g. p, link, noscript, h1, h2, h3 etc. (iii) count of other tags present on both sides changing drastically.

Div Tags: The total division tags increased from 232,054 in original pages to 396,888 in transformed pages. This is attributed to the fact that several other tags such has h1,h2 or p etc. are mapped to div in transformed pages. This can also be visually from the CDFs plotted in Figure 10 and 12. The ratio of division tags to total tags is significantly higher, around 40 to 50 % on average, in transformed pages as compared to around 15 to 20% in original pages.

Image Tags: the image tags also followed a similar pattern as the division tags and increase from 37,544 in original pages to 50,833 in transformed pages. This is again due to the reason that different kind of tags such img, image, cmp-img are mapped to img tag only in transformed pages.

UL Tags: the ul tags decreased drastically from 15,773 to 271 in transformed pages due to changes in page formatting and page layout.

Script Tags: the script tags also decreased from 20,591 to 11,823 in transformed pages due to removal to certain JavaScript content in the transformed pages.

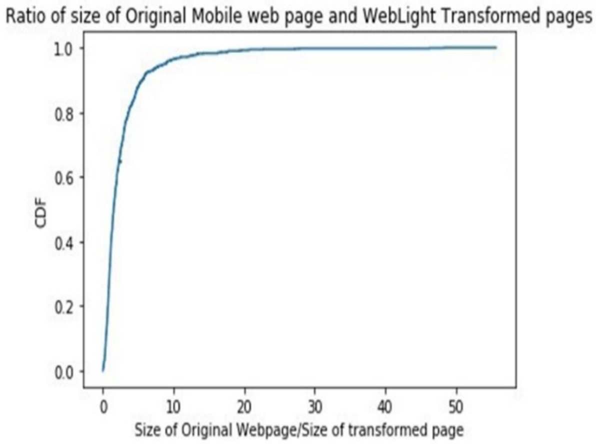


Figure 13: CDF of ratio of original mobile page sizes to transcoded page sizes

5 DEEP LEARNING MODEL

The aim of creating a machine learning model was to learn transformations performed by WebLight using deep learning techniques and then be able to predict these transformations on any other page. Important part before moving towards deciding on deep learning technique was to decide the granularity in which we want to make this model. To get started, we ran an RNN based Natural Language Processing model on raw HTML pages but as expected it did not yield positive results. This meant that we needed to cut down on the complexity. We reduced the complexity by reducing attributes from tag and learning only tags and data encapsulated in these tags. It is important to note that we need to learn the tags in a sequence. We achieve this by deploying a technique that we will explain below. Based on these sequences, we train out Encoder-Decoder model and then translate learned sequences back into an HTML page.

5.1 Web Page to tag sequence via Hierarchical Tag parsing

HTML has a tree-like structure, to learn this structure we can train our model on sequences of tags leading to data such as text. We get these sequences of tags using an approach we came up with. We start parsing HTML file from start, on encountering any start tag enclosed in angular brackets we push them in a stack. Whenever we encounter data while parsing, we copy the image of stack at that time and store it in a global map against that text. We pop from stack whenever we encounter an end tag enclosed in angular brackets and a back slash. Another motivation for going with approach of tag sequence matching was to start by learning features that were common in both original and transcoded pages. One such feature was actual text displayed on both pages. We can leverage this to study patterns across original and transcoded page. Moreover, as fore-mentioned, we have seen how different tags from original pages were being mapped to a limited number of tags in transcoded pages. Therefore, learning these sequences can be very meaningful in context of problem in hand.

5.2 Encoder-Decoder Model

We get tag sequences for both original page and transcoded page in our training dataset and we use these sequences to train our model. Since, now we must map one set of sequence to another, we use an RNN based model. We have used an Encoder-Decoder model that has recently achieved significant results on complex problems like machine translation and video captioning. The most powerful thing about this model is that it can map sequences of different lengths to each other [8]. This is ideal for our use case, since we usually have different size sequences in original and transcoded page.

We feed to our model mappings of sequences from original page to transcoded page. We get these mappings by following steps: Firstly, we run Hierarchical tag parsing algorithm on original page and transcoded page to get tag sequences against text elements in both pages. Next, we compare text elements in original and transcoded page to find mappings by comparing text strings up to a certain index of similarity. To cater for false positives, we only consider text strings greater than 50 and having similarity index of more than 90%. From our dataset, we extracted around 14000 mappings from 1284 pairs of pages. With these mappings as input, batch size of 512, 30 epochs and a validation split of 20% we got an accuracy of 92%. Figure 14 shows accuracy versus epoch and figure 15 shows loss versus epochs graph for our model with fore-mentioned parameters.

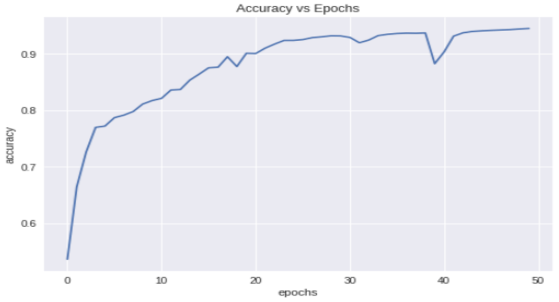


Figure 14: A page transcoded through Google Weblight (https://googleweblight.com/i?u=https://yahoo.com).

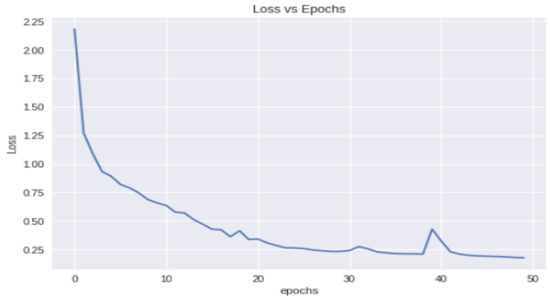
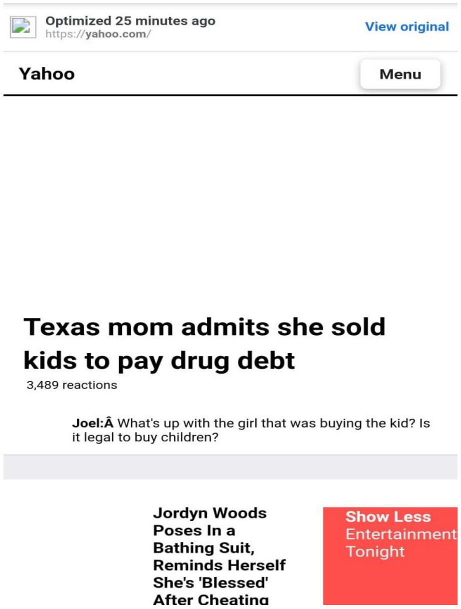


Figure 15: A page transcoded through Google Weblight (https://googleweblight.com/i?u=https://yahoo.com).

Table 1: Comparsion of most frequent tags

Tag name	A Frequency in Original pages	Frequency in transcoded pages
div	232054	396888
a	173996	188058
span	97268	151102
img	37544	50833
script	20591	11823
table	572	12537
h3	10487	0
p	20415	0
path	15469	0



Texas mom admits she sold kids to pay drug debt

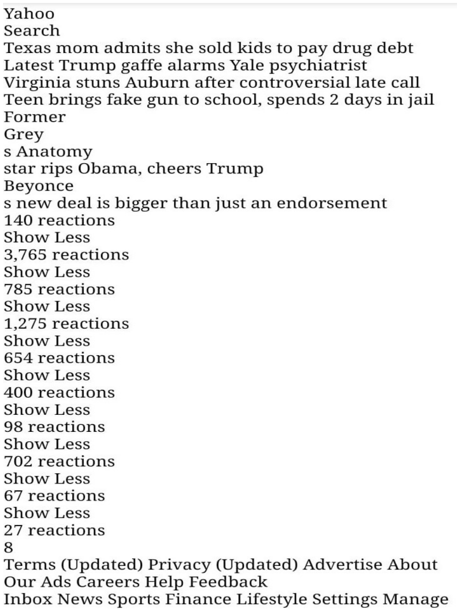
3,489 reactions

Joel:Â What's up with the girl that was buying the kid? Is it legal to buy children?

Jordyn Woods Poses In a Bathing Suit, Reminds Herself She's 'Blessed' After Cheating

Show Less Entertainment Tonight

Figure 16: A page transcoded through Google Weblight (https://googleweblight.com/i?u=https://yahoo.com).



```
Yahoo
Search
Texas mom admits she sold kids to pay drug debt
Latest Trump gaffe alarms Yale psychiatrist
Virginia stuns Auburn after controversial late call
Teen brings fake gun to school, spends 2 days in jail
Former
Grey
s Anatomy
star rips Obama, cheers Trump
Beyonce
s new deal is bigger than just an endorsement
140 reactions
Show Less
3,765 reactions
Show Less
785 reactions
Show Less
1,275 reactions
Show Less
654 reactions
Show Less
400 reactions
Show Less
98 reactions
Show Less
702 reactions
Show Less
67 reactions
Show Less
27 reactions
8
Terms (Updated) Privacy (Updated) Advertise About
Our Ads Careers Help Feedback
Inbox News Sports Finance Lifestyle Settings Manage
```

Figure 17: Same page predicted through our model upto the granularity of tags around texts.

More useful results and sample runs for a few examples have been provided in notebook on our GitHub.

5.3 Reverse Engineering: Tag Hierarchy to HTML page

Using our model, we can now predict tag sequences in transcoded page using tag sequence from original page. We can use sequences for all the text elements to construct an HTML page. We achieve this by constructing a tree for each text field based on its predicted sequence. Then we merge all trees to form one tree and a simple depth first traversal of tree then produces our output HTML file. Since we are not considering any attribute and only considering text fields right now, we do not expect a page exactly similar to transcoded page. However, if we skim transcoded page of additional attributes and images, we will get a similar page. Fig 16 and 17 shows transcoded page and page predicted using our model.

6 FUTURE WORK

Adding more granularity: the next step is to bring in more granularity by considering images as data entities instead of tags. This will greatly enhance the result from Fig x. Moreover, since we have yielded positive results from the model at tag level, we expect it to perform by considering a fewer number of attributes considered as well. Additionally, our approach currently attempts to see relation between data that is present in both the original and the transcoded page. We can further extend our work to incorporate the removal and addition of data as well. Finally, after learning these transformations we aim to suggest improvements in the process of minification that is undertaken by Google WebLight.

7 CONCLUSION

We started by analyzing WebLight pages in multiple domains. Our analysis results brought forward an appreciable number of meaningful insights into how WebLight transforms web pages. It was found that a WebLight transcoded page fetches content only from WebLight servers and a couple of other Google services, compared to 6 non-origin servers for original pages on median. Furthermore, information was gathered to indicate that the transcoded pages only fetch HTML files and exclude CSS and JavaScript files. Additionally, structure within the HTML page also changes drastically, for instance, division tags are converted to double. This gathering of data was followed by the focus put into the development of a machine learning model to predict these transformations. Bypassing technical jargon-we considered tag sequences against text elements from HTML page extracted using an algorithm that we developed. These sequences were used to train a model that had an impressive accuracy of 92%. Lastly, a route to revert back from tag sequences to the actual HTML page was developed. In the future, we aim to invest further effort into the research and consequently extend our implementation to cover the grounds of even a larger variety of cases.

8 REFERENCES

[1] Ahmad, S., Haamid, A. L., Qazi, Z. A., Zhou, Z., Benson, T., & Qazi, I. A. (2016, November). A view from the other side: Understanding mobile phone characteristics in the developing world. InâĂŕProceedings of the 2016 Internet Measurement ConferenceâĂŕ(pp. 319-325). ACM.

[2] Butkiewicz, M., Madhyastha, H. V., & Sekar, V. (2011, November). Understanding website complexity: measurements, metrics, and implications. InâĂŕProceedings of the 2011 ACM SIGCOMM conference on Internet measurement conferenceâĂŕ(pp. 313-328). ACM.

[3] Ling, C., Wang, L., Lang, J., Xia, Q., Chang, G., Wang, K., & Zhao, P. (2018, April). LinCa: A Page Loading Time Optimization Approach for Users Subject to Internet Access Restriction. InâĂŕCompanion of the The Web Conference 2018 on The Web Conference 2018âĂŕ(pp. 69-70). International World Wide Web Conferences Steering Committee

[4] Wang, X. S., Shen, H., & Wetherall, D. (2013, August). Accelerating the mobile web with selective offloading. InâĂŕProceedings of the second ACM SIGCOMM workshop on Mobile cloud computingâĂŕ(pp. 45-50). ACM.

[5] Web Light: Faster and lighter mobile pages from search - Search Console Help. (n.d.).

[6] Wu, O., Hu, W., & Shi, L. (2013). Measuring the visual complexities of web pages.âĂŕACM Transactions on the Web (TWEB),âĂŕ7(1), 1.

[7] Zohar, E., & Cassuto, Y. (2014). Automatic and dynamic configuration of data compression for web servers. InâĂŕ28th Large Installation System Administration Conference (LISA14)(pp. 106-117).

[8] 1143621442379174. (2019, February 05). Understanding Encoder-Decoder Sequence to Sequence Model. Retrieved from https://towardsdatascience.com/understanding-encoder-decoder-sequence -to-sequence-model-679e04af4346

[9] https://github.com/ammartahir24/Web-Light-Analysis

[10] https://googleweblight.com/