# Responsive Text Summarization

Luis A. Leiva*
Sciling, SL
46120 Valencia, Spain

**Abstract**

Responsive text summarization (RTS) is an approach to web design aimed at allowing desktop web pages to be read in response to the size of the device a user is browsing with. RTS implements the TextRank algorithm, so it can be exploited to generate very short summaries (more apt for mobile devices, where screen space is at a premium) or longer, more explicative summaries (more apt e.g. for phablets or laptops). We validate the feasibility of RTS on blog sites and show that runtime performance incurred by our current implementation is negligible. We also show that reading time can be reduced by a factor of 4 without impacting substantially the summary quality. In general, anyone interested in making their web contents concise but informative could benefit from this work.

**Keywords:** Responsive Web Design; Media Queries; Text Summarization; Graph algorithms; Skimming

## 1 Introduction

It is said that, on the web, content is king. However, users often do not read web pages; instead they scan them, trying to pick out a few sentences or even parts thereof to get the information they want [10, 24]. Yet the amount of information available online is growing at an incredible rate. For example, 70 million new posts are created every month on WordPress alone[1] and Facebook users share about 2.5 million pieces of content every minute [28]. Clearly, nobody has time to read everything, yet we often have to make critical decisions based on what we are able to assimilate [21]. Text summarization is becoming indispensable for dealing with this problem, by distilling the most important information from a text source to produce a shorter, more manageable version. However, web designers do not have the right tools to achieve this goal at present. Rather, they usually delegate the implementation backbone to some skilled developer.

Responsive Web Design (RWD) has become a crucial methodology for web designers, as the amount of mobile traffic now accounts for more than half of total Internet traffic,[2] web browsing and apps combined. RWD suggests that design

*Email: `llt@acm.org`

**Figure 1:** An overview of the RTS technique working on different devices. Text sentences are depicted in different colors.

and development should respond to the user's behavior and environment based on screen size, platform, and orientation [15]. The practice consists of a mix of flexible grids and layouts, images, and an intelligent use of CSS media queries that can reformat web pages effortlessly (or automatically) without changing the original content itself or the HTML markup. This would eliminate the need for a different design for each new browser-capable gadget on the market. Yet RWD is primarily targeted at modifying the *graphical* aspect of web pages rather than their *contents*. This work aims at bridging this gap. Responsive text summarization (RTS) is an approach to web design aimed at allowing desktop web pages to be read in response to the size of the device a user is browsing with (Figure 1).

In this paper, we make the following contributions. We introduce a new CSS property that allows web designers to summarize text content easily. Next, we describe our current implementation and evaluate it on blog sites. Finally, we discuss possible directions for future work and thereafter conclude.

## 2 Related Work

There is a large body of work on text summarization techniques that will not be discussed here because space precludes a full treatment. The reader is redirected to comprehensive surveys in this area [1, 9, 11, 13, 23, 26, 29]. Instead, we focus on the RTS idea and the related research by which it was inspired.

### 2.1 Web Content Summarization

RTS relates to previous attempts to use text summarization for web browsing on handheld devices. For example, Lam and Baudisch [17] used text reduction heuristics to create readable overviews, and Buyukkokten et al. [8] investigated different methods for summarizing parts of web pages. Among other findings, text summaries provided significant improvements in terms of time and scrolls required to access the contents. Yu and Miller [30] used human-rated data to make non-salient sentences transparent, aimed at enabling skim-read for non-native speakers.

RTS also relates to the idea of "content magnification" or "semantic zoom" on desktop computers. In this regard, Harrison and Dey [12] took into account people's natural behavior of leaning towards a screen when they need to inspect

unseen details to automatically magnify screen content in proportion to the extent of the lean. Sukale et al. [27] followed this approach and devised the Proxemic Web, where existing information is displayed at different levels of detail depending on the user's distance to the screen.

### 2.2 Web Content Adaptation

Prior to the RWD shift, researchers have proposed different ways of adapting web content, mostly involving end-user interaction. For example, Bolin and co-authors [7] developed a customization plugin for the Firefox browser that enabled a high-level scripting language, without having to access the source code of web pages. Kurniawan et al. [16] proposed to override the visual layer of a web page with custom CSS, although such updates had to be performed by hand. Baudisch et al. [5] and Bila et al. [6] encouraged the user to actively modify the layout contents. We can find even some browser tools to adjust web pages according to specific user needs, e.g. Greasemonkey[3] or Platypus.[4] But having to require the end-user intervention is costly, and will reduce the chance to use these tools when reading web pages. Going further, Leiva [18, 19] leveraged implicit interactions to automatically modify the visual appearance of web page elements. None of these works, however, are tailored to web designers.

### 2.3 Making Text Responsive

The closest attempt to responsive text summarization (as devised in this paper) was a proof of concept which consisted in styling text paragraphs manually, by wrapping "unimportant" words or sentence parts in `<span>` elements that would be hidden on smaller screens.[5] This manual work is clearly too tedious to work at scale, since the text source has to be highly structured in order for it to be readable when parts of it are hidden. On the contrary, our approach allows designers to achieve this goal effortlessly, by simply applying CSS selectors and media queries, as they would typically do in RWD. Some use examples are given in Figure 2.

| Name | Values | Initial value | Inherited? | Media groups |
|---|---|---|---|---|
| `text-summary` | none \| inherited \| <percentage> \| <number> \| <string> | none | no | visual |

**Table 1:** CSS specification proposal (informative, non-normative) of RTS.

## 3 Specification

Most current text summarizers are *extractive* rather than *abstractive*, i.e., they extract and present pieces of the original text rather than rephrase or rewrite them. In RWD it has more sense to use an extractive method for RTS, otherwise the web page could become confusing to the users, as it has been shown that they

generally prefer seeing summarized text in the form that the author created [2]. Thus, RTS presents the user with an extract of the original text. Concretely, an HTML element's text content is replaced by such an extract.

### 3.1 Implementation

At a higher level, RTS provides designers with the `text-summary` CSS property, whose value is either a ratio of the original text length or the number of desired sentences. Table 1 depicts the specification of such property. At a lower level, being an experimental technology, RTS relies on JavaScript to work. Currently, a script file provides the polyfill (browser fallback) that enables RTS in modern browsers. Figure 2 shows some use examples.

```
/* CSS selector. Targets any browser. */
article .content {
  --text-summary: 0.8;
}

/* CSS media query. Targets mobile browsers. */
@media only screen and (max-device-width: 480px) {
  article .content {
    --text-summary: 50%;
  }
}

/* Targets any device in portrait mode. */
@media only screen and (orientation: portrait) {
  article .content {
    --text-summary: 3 sentences;
  }
}
```

**Figure 2:** Working RTS use examples. Since the `text-summary` property is non-standard, it must be prefixed with two dashes (`--`) in order to be recognized by the browser's built-in CSS parser; otherwise it would be ignored.

When the `text-summary` property is defined as a percentage, it may have any value in the range $[0, 1]$ or $[0\%, 100\%]$. Any values outside this range will be clamped. A value of `0.0` implies that the resulting text is a summary of length 0% of the original text, which would result in hiding the whole text source. A value of `1.0` implies that the resulting text is a summary of length 100% of the original text, which would result in leaving the whole text source untouched. Therefore, practical working values would lay in the $[0.01, 0.99]$ or $[1\%, 99\%]$ ranges. In the Evaluation section we provide more insights in this regard. Finally, when the `text-summary` property is defined as a string (e.g. `2 sentences`), it may have any value greater than zero. Otherwise, it is set to `none`.

### 3.2 Algorithm

RTS implements the TextRank summarization algorithm [22], which is a graph-based ranking model. Certainly, there are many extractive summarization algorithms available. We chose TextRank for three main reasons: it is unsupervised, language-independent, and produces results very similar to what a human would produce [22].

TextRank starts by building a graph that represents the text, and interconnects words, sentences, or other text entities with meaningful relations. Let $G = (\mathcal{V}, \mathcal{E})$ be a directed graph with a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. For a given vertex $v_i$, let $\mathcal{D}^-(v_i)$ be the set of vertices that point to it (indegree), and let $\mathcal{D}^+(v_i)$ be the set of vertices that vertex $v_i$ points to (outdegree). The score of such vertex is recursively defined as

$$\mathrm{sc}(v_i) = (1-d) + d \sum_{j \in \mathcal{D}^-(v_i)} \frac{1}{|\mathcal{D}^+(v_j)|} \mathrm{sc}(v_j) \tag{1}$$

where $d$ is a damping factor, usually set to 0.85, as in Google's PageRank algorithm. The derivation of Eq. (1) for weighted graphs is straightforward [22].

In RTS, the goal is to rank entire sentences, and therefore a vertex is added to the graph for each sentence in the text source. To establish connections (edges) between sentences (vertices), we define a similarity relation as a function of content overlap. Formally, given two sentences $\mathcal{S}_i$ and $\mathcal{S}_j$, with a sentence being represented by a set of words $\mathcal{S}_i = \{w_1 \cdots w_i\}$, $|\mathcal{S}_i| \geq 1$ and $|\mathcal{S}_j| \geq 1$, the similarity between them is defined as

$$\mathrm{sim}(\mathcal{S}_i, \mathcal{S}_j) = \frac{|\{w_k : w_k \in \mathcal{S}_i \bigcap \mathcal{S}_j\}|}{\log |\mathcal{S}_i| + \log |\mathcal{S}_j|} \tag{2}$$

All the similarity scores can be arranged in an adjacency matrix $\mathbf{A}_{|\mathcal{V}| \times |\mathcal{V}|}$, where rows and columns are the sentences to rank. Let $a_{ij} = \mathrm{sim}(\mathcal{S}_i, \mathcal{S}_j)$ be an element of $\mathbf{A}$. The indegree and outdegree of vertex $v_i$ are computed as follows:

$$\begin{aligned} \mathcal{D}^-(v_i) &= \sum_j a_{ij} \\ \mathcal{D}^+(v_i) &= \sum_j a_{ji} \end{aligned} \tag{3}$$

Finally, the top N ranked sentences makes up the text summary, as indicated by the web designer via the `text-summary` CSS property. This way, if the text source has e.g. 10 sentences and `text-summary` is set to 0.6, then the top 6 sentences of the original text would be the summary.

A very important advantage of rank-based algorithms such as TextRank is that they prioritize all sentences in a text, which means that they can be exploited to extract very short summaries (more apt for mobile devices, where screen space is at a premium) or longer, more explicative summaries (more apt e.g. for phablets or laptops). In particular, TextRank works well because it does

not only rely on the local context of a text unit (vertex), but rather it takes into account information recursively drawn from the entire text (graph).

To conclude this section, we should mention that RTS preserves the original HTML tags in the generated summaries. This is accomplished via a simple string matching algorithm. First, sentences in the text source are matched in plain text against each sentence in the summary. Next, the summary sentences are replaced by the original sentences from the text source.

# 4 Evaluation

Aimed at gaining insights about the RTS technique, we conducted three experiments. The interested reader may consult recent qualitative and quantitative evaluations of several state-of-the-art text summarizers elsewhere [3, 4].

## 4.1 Method

In the first experiment, we evaluate the impact on runtime performance incurred by RTS. In the second experiment, we analyze the impact of different summary ratios on reading time, for which we employ the usual approximation of 250 words per minute [25]. In the third experiment we analyze the impact of different summary ratios on summary quality, for which we adopt the traditional Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metric [20]. ROUGE takes into account the number of overlapping text units between a computer-generated summary and the ideal summary created by human labelers. In RTS, the main text units are sentences; therefore we will use the ROUGE-L variant. ROUGE-L looks at the longest common subsequence between two strings (in-sequence co-occurrences). ROUGE-L is 1.0 when the summary perfectly matches any of their references.

## 4.2 Data

Notice that having to conduct manual comparison of peer summaries with different summary ratios is an arduous and costly process. Luckily, a body of research has been produced in the last decade to benchmark different text summarization systems. Therefore, we will use the dataset provided by Hu et al. [14] as gold standard, since it is publicly and freely available.

The dataset comprises 100 blog posts annotated at the sentence level; see Table 2. Four human labelers were asked to provide an extractive summary of each blog post; see Table 3. This way, each sentence in each blog post was annotated as relevant or non-relevant by each labeler. On average, each labeler deemed as relevant 610 sentences (17,100 words) overall. Therefore, human labelers produced summaries with 28.7% less text on average, which would account for a summary ratio of near 70% of each blog post.

| No. text sources | No. sentences | No. words |
|---|---|---|
| 100 | 2,122 | 41,563 |

**Table 2:** Descriptive statistics of the dataset.

| Human labeler ID | No. sentences | No. words |
|---|---|---|
| 1 | 609 | 17,447 |
| 2 | 640 | 16,934 |
| 3 | 634 | 17,817 |
| 4 | 555 | 16,212 |

**Table 3:** Statistics of the human labelers.

### 4.3 Results

To begin, in the first experiment we get runtime performance estimates incurred by our current implementation for a summary ratio of 50%, which represents a trade-off between all the possible values in the working RTS range; see Implementation section. We analyzed runtime performance on three different devices: a desktop computer (i7 CPU@3.3 GHz), a laptop (i5 CPU@1.6 GHz), and a mobile device (Snapdragon CPU@1 GHz). We used the latest version of Google Chrome in all devices. The number of test runs was set to $N = 1000$ iterations.

Unfortunately, the source URLs of the blog posts are not available. Therefore, we generate a basic HTML page of each blog post with a `<div>` element that is populated with the post content. Then, we add a stylesheet with a CSS selector that summarizes the text content to 50%. We also add our polyfill together with a custom script that computes the time required both to parse the stylesheet and to perform text summarization itself. As can be observed in Table 4, the impact on runtime performance is negligible for any of the tested devices. This suggests that RTS is a very feasible approach. In general, we have observed that the performance results for other summary ratios are within the same order of magnitude.

| Device | CSS parsing (ms) | Summarization (ms) |
|---|---|---|
| Desktop | 0.03 ± 0.01 | 2.12 ± 0.15 |
| Laptop | 0.08 ± 0.01 | 12.26 ± 2.48 |
| Mobile | 0.29 ± 0.03 | 35.25 ± 6.33 |

**Table 4:** Performance results (mean ± sd processing times) for a summary ratio of 50%.

The results of the second and third experiments are reported jointly in Figure 3. As expected, reading time grows linearly with the summary ratio. Interestingly, there is a sweet spot in summary quality attributed to a summary ratio of 70%, for which the ROUGE score is maximum ($M$=0.71, $SD$=0.14). This

is actually in line with what human labelers produced, as discussed previously. For higher and lower summary ratios around such sweet spot, the summary quality degrades slightly. However, the differences between summary ratios of 30% onward are not statistically significant, as revealed by post-hoc pairwise comparisons of proportions (Bonferroni corrected) after the chi-square test ($\chi^2_{(8,N=100)} = 22.84, p < .01$). It was for ratios between 10% and 20% where those differences in summarization quality were found to be statistically significant ($p < .01$).

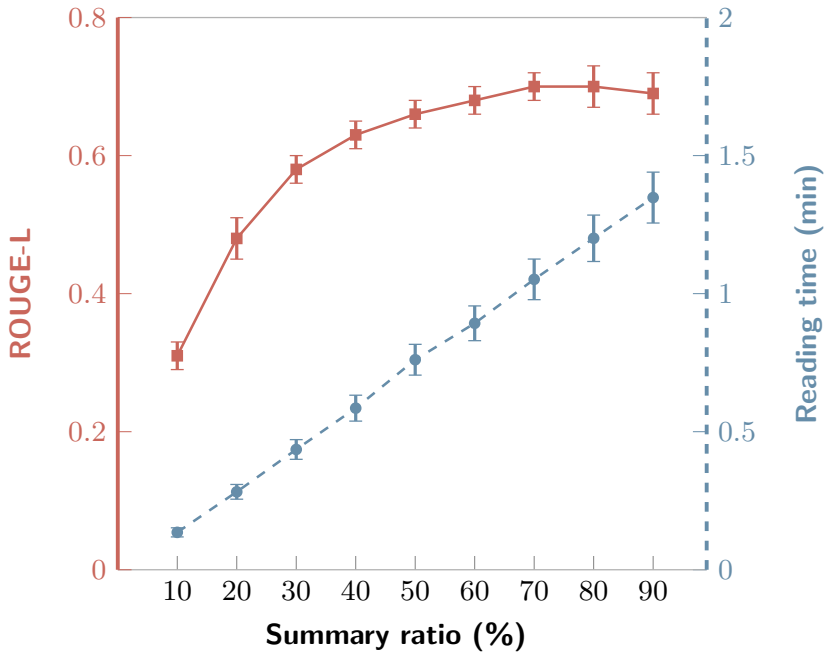


**Figure 3:** ROUGE scores and reading times *vs.* summary ratio. Error bars denote 95% confidence intervals.

In sum, RTS provided similar ROUGE scores for summary ratios of 30% onward. This means that we can expect that a reduction of 70% in blog post length will not impact substantially the summary quality. Further, this reduction accounts for reading time savings of 73.85% per blog post on average. For example, a blog post that takes 10 min to read would take 2.6 min to read with `text-summary:30%`. Taken together, our results suggest that RTS succeeds efficiently in providing concise but informative summaries.

Anecdotally, we should mention that RTS outperforms state-of-the-art results in the same summarization task over the same dataset; c.f. [14]. This suggests that RTS, despite using a relatively straightforward summarization algorithm, may perform really well in practice.

## 5 Conclusion and Future Work

RTS is an experimental technology for RWD that allows desktop web pages to be read in response to the size of the device a user is browsing with. With RTS, web designers can create custom reading solutions for a wide range of users, on a wide range of devices. RTS might be especially useful for news or blog sites, as a means to quickly allow users keeping up on the latest events.

We plan to escalate the RTS specification to the W3C CSS working group, since it can be easily implemented in modern browsers. Another research avenue is the client-side generation of summary snippets, so that they can be inserted at the discretion of web content creators.

Our software is publicly available at `https://luis.leiva.name/rts/`. In general, anyone interested in making their websites concise but informative could benefit from this work. In conclusion, we feel that RTS has great potential, and the work presented here opens important opportunities for future efforts in this area.

## Funding

This research did not receive any funding from the public, commercial, or not-for-profit sectors.

## Endnotes

[1] `https://wordpress.com/activity/`
[2] `http://www.comscore.com/USMobileAppReport`
[3] `http://greasemonkey.mozdev.org`
[4] `http://platypus.mozdev.org`
[5] `http://www.frankieroberto.com/responsive_text`

## References

[1] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut. Text summarization techniques: A brief survey. In *Proc. Computing Research Repository*, CoRR, pages 1–9, 2017.

[2] Aris Anagnostopoulos, Andrei Z. Broder, Evgeniy Gabrilovich, Vanja Josifovski, and Lance Riedel. Web page summarization for just-in-time contextual advertising. *ACM Trans. Intell. Syst. Technol.*, 3(1):14:1–14:32, 2011.

[3] Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. Variations of the similarity function of textrank for automated summarization. In *Proc. Argentine Symposium on Artificial Intelligence*, ASAI, pages 65–72, 2015.

[4] Jamilson Batista, Rodolfo Ferreira, Hilário Tomaz, Rafael Ferreira, Rafael Dueire Lins, Steven Simske, Gabriel Silva, and Marcelo Riss. A quantitative and qualitative assessment of automatic text summarization systems. In *Proc. ACM Symposium on Document Engineering*, DocEng '15, pages 65–68, 2015.

[5] Patrick Baudisch, Xing Xie, Chong Wang, and Wei-Ying Ma. Collapse-to-zoom: Viewing web pages on small screen devices by interactively removing irrelevant content. In *Proc. ACM Symposium on User Interface Software and Technology*, UIST '04, pages 91–94, 2004.

[6] Nilton Bila, Troy Ronda, Iqbal Mohomed, Khai N. Truong, and Eyal de Lara. PageTailor: Reusable end-user customization for the mobile web. In *Proc. Intl. Conf. on Mobile Systems, Applications and Services*, MobiSys '07, pages 16–29, 2007.

[7] Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. Automation and customization of rendered web pages. In *Proc. ACM Symposium on User Interface Software and Technology*, UIST '05, pages 163–172, 2005.

[8] Orkut Buyukkokten, Hector Garcia-Molina, and Andreas Paepcke. Seeing the whole in parts: Text summarization for web browsing on handheld devices. In *Proc. Intl. Conf. on World Wide Web*, WWW '01, pages 652–662, 2001.

[9] Dipanjan Das and André F. T. Martins. A survey on automatic text summarization. Technical report, Carnegie Mellon University, 2007.

[10] Geoffrey B. Duggan and Stephen J. Payne. Skim reading by satisficing: Evidence from eye tracking. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, CHI '11, pages 1141–1150, 2011.

[11] Mahak Gambhir and Vishal Gupta. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66, 2017.

[12] Chris Harrison and Anind K. Dey. Lean and zoom: Proximity-aware user interface and content magnification. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, CHI '08, pages 507–510, 2008.

[13] Eduard Hovy. Text summarization. In Ruslan Mitkov, editor, *The Oxford Handbook of Computational Linguistics*. 2005.

[14] Meishan Hu, Aixin Sun, and Ee-Peng Lim. Comments-oriented document summarization: Understanding documents with readers' feedback. In *Proc. ACM SIGIR Conf. on Research and Development in Information Retrieval*, SIGIR '08, pages 291–298, 2008.

[15] Kayla Knight. Responsive Web Design: What it is and how to use it. Smashing Magazine, 2011. Available at `https://smashingmagazine.com`.

[16] S. H. Kurniawan, A. King, D. G. Evans, and P. L. Blenkhorn. Personalising web page presentation for older people. *Interact. Comput.*, 18(3):457–477, 2006.

[17] Heidi Lam and Patrick Baudisch. Summary thumbnails: Readable overviews for small screen web browsers. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, CHI '05, pages 681–690, 2005.

[18] Luis A. Leiva. Restyling website design via touch-based interactions. In *Proc. Intl. Conf. on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pages 599–604, 2011.

[19] Luis A. Leiva. Automatic web design refinements based on collective user behavior. In *Proc. Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 1607–1612, 2012.

[20] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Proc. ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July 2004.

[21] Inderjeet Mani. *Advances in Automatic Text Summarization*. MIT Press, 1999.

[22] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into texts. In *Proc. Empirical Methods in Natural Language Processing*, EMNLP '04, pages 404–411, 2004.

[23] Ani Nenkova. A survey of text summarization techniques. In C.C. Aggarwal and C.X. Zhai, editors, *Mining Text Data*. 2012.

[24] Jakob Nielsen and John Morkes. Concise, SCANNABLE, and Objective: How to write for the Web. Nielsen Norman Group, 1997. Available at `https://nngroup.com`.

[25] K. Rayner, E. R. Schotter, M. E. J. Masson, M. C. Potter, and R. Treiman. So much to read, so little time: How do we read, and can speed reading help? *Psychological Science in the Public Interest*, 17(1):4–34, May 2016.

[26] Chintan Shah and Anjali Jivani. Literature study on multi-document text summarization techniques. In *Proc. Intl. Conf. on Smart Trends for Information Technology and Computer Communications*, SmartCom, pages 442–451, 2016.

[27] Ryan Sukale, Olesia Koval, and Stephen Voida. The proxemic web: Designing for proxemic interactions with responsive web design. In *Proc. ACM Intl. Joint Conf. on Pervasive and Ubiquitous Computing: Adjunct Publication*, UbiComp '14 Adjunct, pages 171–174, 2014.

[28] Jake Taylor. 211 million: This is how much online content is created every minute, 2015. Available at `http://littlejackmarketing.com`.

[29] Jaya Kumar Yogan, Ong Sing Goh, Basiron Halizah, Hea Choon Ngo, and C. Suppiah Puspalata. A review on automatic text summarization approaches. *J. Computer Science*, 12(4):178–190, 2016.

[30] Chen-Hsiang Yu and Robert C. Miller. Enhancing web page skimmability. In *Proc. Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 2655–2660, 2012.