July 05, 2018

Automatic loading of document content

n/a

Automatic loading of document content

ABSTRACT

Web pages can use dynamic content loading technology to facilitate viewing on mobile devices. This disclosure describes techniques in which a computer system can automatically initiate loading of additional document content that is not served as part of the initial loading of a document. The computer system can identify and activate controls in the document, such as a "show more" button, that cause additional content to be requested and inserted into the document, initiate scrolling operations that trigger loading of additional content into the document, etc. Described techniques allow a device to automatically and efficiently obtain a more complete version of a document, which can improve document indexing, document transcoding, crawling of web pages, and other processes.

KEYWORDS

- web page
- display content
- load content
- automatic trigger
- scroll

BACKGROUND

With the development of web-based technology, modern web pages can use dynamic content loading technology to facilitate viewing on mobile devices. For example, for some web pages are configured so that a mobile device only loads a portion of the content of the web page (e.g., the "main content") at first. The remaining content is "folded" (e.g., hidden, or not loaded) to reduce latency of the initial loading and conserve bandwidth for the mobile device.

The remaining content that was initially "folded" or "hidden" can be provided after interaction by a user with the webpage, e.g., such interactions as scrolling of the web page, or clicking a user interface (UI) button such as a "see details" or a "show more" button. Such required user interactions can hinder operations that would take advantage of the full content of the document, e.g., operations such as indexing for search engines or transcoding of a document.

DESCRIPTION

This disclosure describes techniques in which a computer system can automatically initiate loading of and obtain additional document content that is not served or transferred as part of the initial loading of a document. The techniques include, without user interaction, the system identifying and activating user interface (UI) controls in the document, such as a "show more" button, that cause additional content to be requested and inserted into the document. Similarly, without user interaction, the system can initiate scrolling operations that trigger loading of additional content into the document. These and other techniques discussed below can allow a server system to automatically and efficiently obtain a more complete version of a document, which can improve document indexing, document transcoding, crawling of web pages, and other processes.

As an example, a computing device may request a web page for further processing, e.g., for indexing or generation of a transcoded version of the web page. The web page can include multiple sections, including an initial section that is provided on initial loading, as well as other sections that are hidden. The hidden sections are not initially included by the web page upon initial loading, but are provided after an event such as scrolling of the web page or activation of an interactive UI element. For example, the interactive UI element can be a "show more" button that triggers additional content to be loaded at the bottom of the web page. The device

can, upon receipt of the web page, automatically scroll the web page without user interaction. The device can also locate the interactive UI element, for example, using a template that corresponds to the Internet domain of the web page. The device, upon identification of the interactive UI element, can activate the interactive UI element without user interaction to initiate loading of the additional content.

The device can, upon receipt of the additional content from a content server, index the additional content – including associating the additional content with an identifier in a document index for the web page. Further, the device can, upon receipt of the additional content from the content server, generate a transcoded version of the web page that includes the initial content and the additional content. The transcoded version may represent a streamlined or compressed version of the page that can be provided to computing devices operating with low communication bandwidth.

Generation and Validation of Templates for Identifying UI Elements

FIG. 1A depicts an example of a system that can generate templates used for the identification of interactive UI elements that are included within electronic documents. These templates can later be used by a computing system to efficiently identify and activate interactive UI elements that cause additional document content to be loaded.

The system of Fig. 1A includes a computing system 102 and a template data store 104. The computing system 102 can include an interactive UI element detector 106, an URL pattern detection module 108, a template generation module 110, a URL sampling module 112, and a URL pattern validation module 114. The computing device 102 can be in communication with the template data store 104, e.g., over one or more networks.

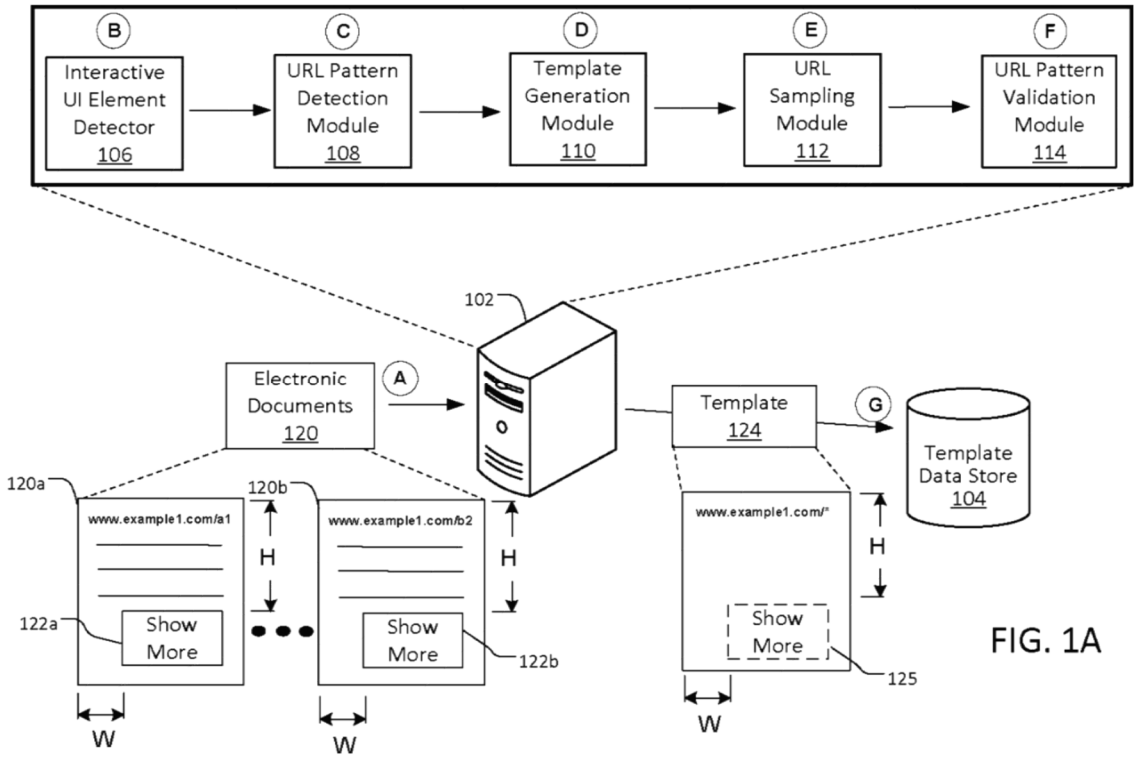

FIG. 1A

Fig. 1A shows a number of steps labeled (A) through (H) to be performed, which illustrate a flow of data.

In step (A), the computing system 102 can receive electronic documents 120, e.g., from a server system or from a data store. The electronic documents can include web pages. Example electronic document 120a can be associated with a web address of "www.example1.com/a1" and the example electronic document 120b can be associated with a web address of "www.example1.com/b2." The electronic documents 120a, 120b can be associated with the same Internet domain, e.g., "www.example1.com."

In step (B), the interactive UI element detector 106 identifies interactive UI elements, of the electronic documents, that trigger loading of additional content into the documents. The interactive UI elements are configured such that activation of the interactive UI elements initiates loading of additional content of the associated electronic document. For example, the interactive UI elements can include a "show more" or a "load more" interactive UI element (e.g., UI button) such that when activated (e.g., selected), loading of additional content of the associated electronic document is initiated. This loading may include, for example, expanding of a list of items in the page, e.g., the insertion of or appending of additional images, search results, social media posts, or other items at the bottom of a list in the page.

For each electronic document, the interactive UI element detector identifies the interactive UI elements by initially identifying a document object model (DOM) element that is included in the electronic document. The interactive UI element detector analyzes the DOM element to identify features of the DOM element, such as a position of the DOM element within the electronic document, a visibility of the DOM element within the electronic document, and a class name of the DOM element. The interactive UI element detector compares the features to respective predetermined reference values, and based on the comparing, identifies the DOM element as an interactive UI element. In this example, the interactive UI element detector identifies the interactive UI elements 122a, 122b of the electronic document 120a, 120b, respectively.

The computing system can use a variety of techniques to determine whether a UI element triggers loading of additional content into a page. For example, the computing system can examine displayed text of a UI element, a location of the UI element, a visibility of the UI element, and a size of the UI element. The computing system can look for elements that

display text known to be associated with additional loading, such as “show more,” “load more,” “expand list,” and the like. The computing system can weight different UI elements according to their position in a document, with UI elements closer to the bottom of the page being weighted as more likely to initiate additional loading. The computing system can determine whether a UI element is within an appropriate range of sizes to be interacted with by a user, e.g., greater than a minimum size and less than a maximum size. The computing system can assess whether a UI element is interactive or not, and which event handlers are associated with the UI element. For example, a UI element to trigger additional loading would generally be expected to have a tap or click event handler to respond to user interaction.

Each of these factors can be used to determine a score, for each of the various UI elements in a document, representing how likely the element is to initiate loading additional content. The computing system can then determine a candidate set of UI elements for initiating additional loading, e.g., by selecting UI elements having scores that satisfy a threshold, or by ranking the UI elements according to the scores and selecting a highest-ranking subset.

The computing system can also evaluate candidate UI elements by initiating events and assessing the results of the events. For example, for each UI element in the set of candidate UI elements, the computing system can initiate a click event to activate the UI element, and then compare the resulting rendered document view with the previous rendered document view. Generally, a control to initiate loading of additional content would add additional content into a page, especially at the bottom of the page, while preserving other content. If activating a UI element navigates to a different URL or removes content of the initial page, the computing system may determine that the UI element does not represent a “show more”-type control to initiate loading of additional content into the page. On the other hand, when the computing

system determines that activating a UI element maintains the current URL, substantially maintains previously loaded content of the document, and adds at least a minimum amount or a certain type of content to the document, the computing system can determine that the UI element represents a control for initiating loading of additional content.

In step (C), the URL pattern detection module 108 can group the electronic documents that include interactive UI elements to initiate loading of additional content. The purpose of the grouping is to group documents that have UI elements to initiate content loading in the generally the same locations. As a result, a single template for locating these UI elements can be used for all of the documents in a group.

The grouping can be done based on characteristics of the respective electronic documents. Since many documents in the same Internet domain have similar layouts and formatting, the URL of a document can be one of the criteria used to group electronic documents. For example, the URL pattern detection module can initially group the electronic documents by Internet domain and other features of the URL. For example, the URL pattern detection module can create a grouping of the electronic documents that are associated with a common Internet domain. In the illustrated example, the URL pattern detection module groups the electronic documents 120a, 120b together based on the electronic documents associated with a common Internet domain – e.g., "www.example1.com." Of course, simply having a common Internet domain does not mean that "show more" controls are located in the same areas across the entire domain. Groups of documents are further refined based on the commonality and consistency of the positions of the UI elements for initiating additional content loading. As a result, multiple different groups may be defined for a single Internet domain, to represent the different types of pages and layouts that may be present. For example,

"www.example1.com/news/*" and "www.example1.com/products/*" may represent different groups of pages, where the pages within each group have a common placement of a "show more" button, but the placement is different from one group to the other.

Furthermore, the groups can be defined so that for each group of the electronic documents, the interactive UI elements to initiate additional content loading have similar properties. For example, for each group of the electronic documents, the interactive UI elements of the respective electronic documents of the group can be associated with a similar position, visibility, class name, or a combination thereof, within the respective electronic document. In the illustrated example, the interactive UI elements 122a, 122b of the group of electronic documents 120a, 120b share similar properties – e.g., a position, a visibility, and/or a class name of each of the interactive UI elements 122a, 122b are substantially similar.

In this example, the URL pattern detection module identifies the position of the interactive UI elements 122a, 122b of the group of electronic documents including the electronic documents 120a, 120b. For example, the position can include a height H spacing of the interactive UI elements 122a, 122b with respect to a top of the respective electronic documents 120a, 120b; and a width W spacing of the interactive UI elements 122a, 122b with respect to a side of the respective electronic documents 120a, 120b. In this example, the height H spacing and the width W spacing associated with interactive UI element 122a, 122b are substantially the same. The position may be determined relative to any portion of a document, e.g., the top, bottom, or sides of the document, or another landmark in the content of the document (e.g., a toolbar, a menu, an image, etc.).

In some examples, the URL pattern detection module 108 can include a CSS selector. Specifically, the CSS selector can utilize pattern matching rules that determine which style rules

apply to the interactive UI elements 122a, 122b of the group of electronic documents including the electronic documents 120a, 120b. If all conditions in a pattern are satisfied, the URL pattern detection module 108 matches the interactive UI elements 122a, 122b to a pattern.

In step (D), the template generation module 110 generates a template 124 for the electronic documents 120a, 120b. For example, the template can identify the interactive UI elements for loading additional content for a particular group of web pages. In this example, the template includes data identifying the locations and other properties of the interactive UI elements 122a, 122b. The computing system determined that the "show more" element is consistently applicable to the pages having the URL pattern "www.example1.com/*", where the asterisk is a wildcard character that matches any text. The template can include data identifying characteristics of the interactive UI elements 122a, 122b associated with the electronic documents 120a, 120b such as a position, a visibility, and/or a class name of the interactive UI elements 122a, 122b. This information allows the template to be used to identify similar interactive UI elements that initiate content loading for pages matching the URL pattern, even if the specific pages have not been analyzed by the computer system.

In some examples, the template designates a position a document where a UI element for initiating additional content loading is expected to occur, e.g., at the height H spacing and width W spacing as associated with the interactive UI elements 122a, 122b. When the template is applied to an electronic document having a URL that matches the URL pattern "www.example1.com/*," e.g., a document hosted in the "www.example1.com" domain, the "show more" button for the document is expected to be located at the location indicated by the template. In some examples, the template specifies data, such as a URL pattern, that indicates which electronic documents should be assessed using the template. For example, the template

can include data specifying the set of documents having URLs that match the URL pattern "www.example1.com/*". From this data, a computing system can determine that the template is appropriate for finding a "show more" button in an electronic document associated with the domain "www.example1.com/," because the URL of the document matches the URL pattern indicated by the template.

In step (E), the URL sampling module 112 can randomly identify and retrieve a sample of the electronic documents that are associated with the previously identified URL pattern. For example, the URL sampling module 112 can select a subset of the electronic documents that are associated with respective URLs that match the URL pattern "www.example1.com/*."

In step (F), from the selected electronic documents having a URL that matches the identified URL pattern, the URL pattern validation module 114 validates the content of the template. For example, the computing system can test whether the selected documents have an interactive UI element to initiate additional content loading at the position indicated by the template. For the selected electronic documents, the interactive UI elements that trigger loading of additional content are identified, e.g., by the URL pattern validation module applying the template for the URL pattern "www.example1.com/*" to the selected electronic documents. For example, the URL pattern validation module applies the template to the electronic document 120a. In response to applying the template to the selected electronic documents, the URL pattern validation module identifies the interactive UI element 122a of the electronic document 120a. Identification of the interactive UI element 122a can include the template interactive UI element 125 aligning with the UI element 122a. For example, a positioning (e.g., height H spacing and width W spacing) of the template interactive UI element

corresponds to the positioning (e.g., height H spacing and width W spacing) of the interactive UI element 122a.

The URL pattern validation module activates the interactive UI element 122a, e.g., by triggering a JavaScript event, to determine if activation of such is successful. For example, activation of the interactive UI element 122a is successful when i) subsequent to the activation, the electronic document 120a includes visible tokens or outlinks greater than a threshold; ii) the activation does not result in token or outlink loss greater than a threshold; and iii) the href (i.e., the URL) of the electronic document 120a is not changed by the activation.

The URL pattern validation module determines, for the template, a success ratio of the selected electronic documents that include interactive UI elements that trigger loading of additional content at the positions indicated by the template. For example, a threshold of the success ratio can be 70% -- that is, if at least 70% of the electronic documents that include interactive UI elements are successfully activated to load more content into a document, the template is maintained. However, if less than 70% of the electronic documents that include interactive UI elements are not successfully activated, the template is not maintained. For example, the URL pattern for the template may be altered to be stricter, e.g., to encompass the subset of documents for which additional content loading occurred but exclude documents that did not have the same layout or “show more” button placement.

In step (G), the computing system stores the template within the template data store 104. For example, when the success ratio of the template is above a threshold, the computing system stores the template in the template data store. As such, when additional electronic documents are received, the template can be applied to the additional electronic documents to identify the interactive UI elements of the additional electronic documents, described further herein.

Loading additional content of documents via interactive UI elements

FIG. 1B illustrates a system for loading additional content of documents. This example illustrates how templates, which have been generated as described above, can be applied to electronic documents for identification of interactive UI elements included in the electronic documents. As shown, the computing system includes a template identification module 150, a scrolling module 152, and a template application module 154.

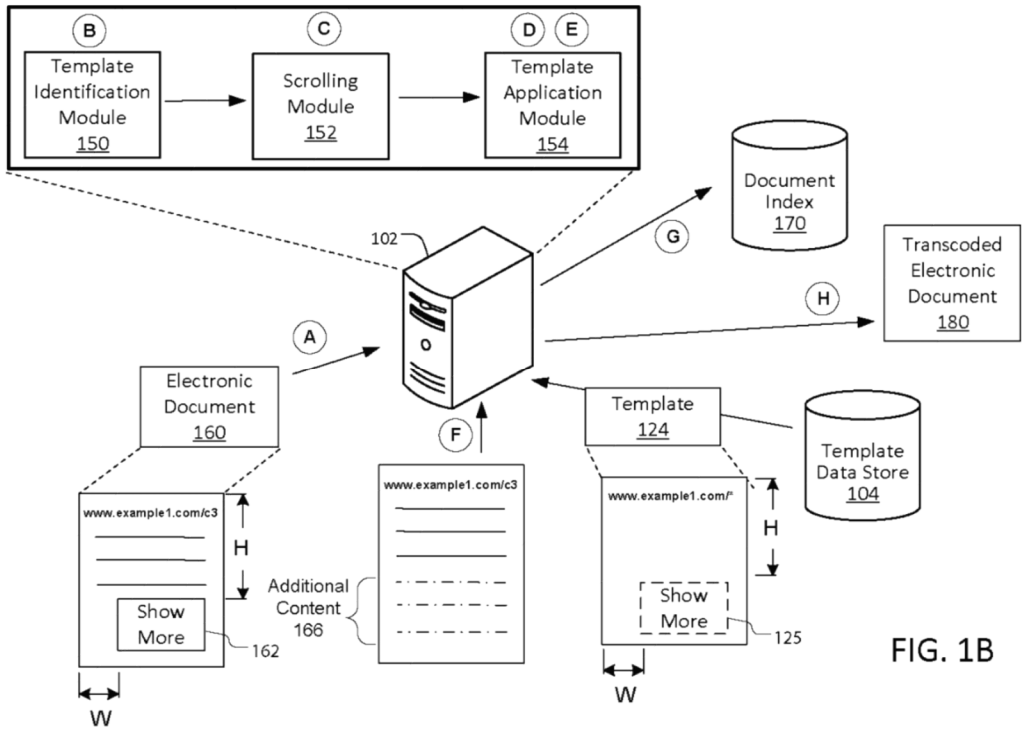

FIG. 1B

Fig. 1B shows a number of steps labeled (A) through (H) to be performed, which illustrate a flow of data.

In step (A), the computing system identifies an electronic document 160. For example, the computing system can receive the electronic document from a server system or from a data store. In some examples, the electronic document includes a web page. In this example, the electronic document is associated with a web address of "www.example1.com/c3."

In some examples, identifying the additional electronic document 160 includes receiving the electronic document at a server system rather than a user device. For example, the computing system can use a "headless" browser, e.g., one that renders documents but does not actually display them visually. In some examples, upon receipt of the electronic document, or shortly thereafter, the computing system can identify parameters of the electronic document. For example, the computing system can identify initial values of a scroll offset, a hypertext reference (href), a title, and a canonical URL associated with the electronic document. In some examples, the computing system stores such initial values in a memory.

In step (B), the template identification module 150 selects a template from multiple templates based on characteristics of the electronic document. For example, the template identification module identifies a URL for the electronic document and identifies a template having a URL pattern that matches the URL for the document. In this example, the template identification module identifies the Internet domain of the electronic document as "www.example1.com." The template identification module accesses the template data store 104 to select a template that is associated with the same Internet domain that is associated with the electronic document. The template data store includes a plurality of templates that are each associated with a different Internet domain. The template identification module can select the

template that is associated with the same Internet domain as associated with the electronic document, e.g., "www.example1.com."

In step (C), the scrolling module 152 scrolls the electronic document. Although the document is not visually rendered and no user is interacting with the rendering, the computing system causes scroll events to be transmitted to cause scrolling to occur. For example, after identifying the electronic document, the scrolling module automatically scrolls the electronic document to an end of the electronic document without any user input, e.g., not in response to user input. The end of the electronic document can include an end (or bottom) of displayable content of the electronic document.

In some examples, the scrolling module scrolls the electronic document by a specific amount – e.g., by a predetermined pixel height, such as a typical pixel height of a display of certain class of devices (e.g., smartphones, tablet computers, etc.). In some examples, the scrolling module scrolls the electronic document 160 by the specific amount multiple times, e.g., scrolls the electronic document by the specific amount, then attempts to scroll the electronic document again by the specific amount or a portion of the specific amount, until scrolling can no longer increase a height of the electronic document. In some examples, such scrolling is performed until a threshold number is reached – e.g., to avoid infinite scrolling of certain electronic documents that allow such. In some examples, the scrolling module scrolls the electronic document until identification of an interactive UI element of the electronic document is facilitated, described further herein.

In some examples, automatically scrolling the electronic document by the scrolling module includes triggering a JavaScript event to scroll the electronic document without user interaction to initiate the scrolling of the electronic document. In some examples, automatically

scrolling the electronic document by the scrolling module is performed prior to applying the template to the electronic document. Scrolling may be performed in several incremental steps, for example, scrolling to the end of current loaded content, waiting for a certain amount of time, and then initiating additional scrolling. Some documents respond to scrolling by automatically loading additional content into the page, and pausing between scroll actions can allow these documents to incrementally load content after each scroll action. If scrolling and pausing for a predetermined amount of time does not result in additional content being added or additional room to scroll being added, the scrolling phase can end, and the further actions discussed below can be performed.

In step (D), the template application module 154 locates in the electronic document an interactive UI element 162 for initiating loading of additional document content. For example, the template application module, using the template identified at step (B), locates the interactive UI element 162. For example, the template can include data identifying an interactive UI element such as a position, a visibility, and/or calls name of the interactive UI element. To that end, the template application module applies the template to the electronic document to identify the interactive UI element 162. In some examples, the template application module is able to locate the interactive UI element 162 based on a position of the interactive UI element 162 with respect to the electronic document being substantially the same as a position the template element 125 with respect to the template 124. That is, a positioning (e.g., height H spacing and width W spacing) of the interactive UI element 162 corresponds to the positioning (e.g., height H spacing and width W spacing) of the template element 125.

In some examples, the template application module is able to locate the interactive UI element 162 representing a “show more” control based on the URL of the document. The

template application module can select the template as being appropriate for the electronic document based on a URL pattern, or a set of URLs, stored by the template. The template application module can determine whether the URL of the document matches the URL pattern of the template and, if there is a match, use the template to locate a "show more" control configured to load additional content into the document. For example, when processing an electronic document associated with the high-level domain "www.example1.com/," the template application module can determine that the template is appropriate since the URL of the electronic document 160 is "www.example1.com/c3," which meets the pattern indicated by the template. The "show more" button for the document can be identified based on the pattern stored by the template.

In step (E), the template application module activates the located interactive UI element 162, without user interaction with the interactive UI element 162. In some examples, the interactive UI element is activated by triggering a JavaScript event, e.g., a click or tap event for the UI element to simulate a user interaction with the UI element. In some examples, after the interactive UI element is activated, the computing system determines whether the initial values of the parameters of the electronic document have changed, such as initial values of the hypertext reference (href), the title, and the canonical URL associated with the electronic document.

In step (F), the computing device receives additional content of the electronic document in response to activation of the interactive UI element 162. For example, the computing device receives additional content 166 of the electronic document over a communications network. In an example, upon activation of the interactive UI element or shortly thereafter, the computing device transmits a network request to a content server (not shown) providing the electronic

document and the additional content. In response, the content server (not shown) transmits the additional content to the computing device.

In some examples, after receiving the additional content of the electronic document, and in response to the computing device determining that the initial values of the parameters of the electronic document have changed, the computing device resets the parameters of the electronic document to the initial values. For example, the computing device can reset the values of the scroll offset, the hypertext reference (href), the title, and the canonical URL associated with the electronic document to respective initial values. In some examples, resetting the parameters of the electronic document to the initial values can include the scrolling module 152 automatically scrolling the electronic document to a position associated with the scroll offset. That is, the scrolling module can automatically scroll the electronic document to an original position of the electronic document, e.g., as initially provided to the computing device at step (A).

The steps (C) through (F) may be repeated to continue to load additional content of the document 160. For example, the computing system can continue to scroll the document through the most recently obtained content, then locate and activate the “show more” button again to load yet additional content. The process may continue until the end of the document is reached or an end of available content of the document (e.g., loadable content from a server) is reached. This process of scrolling and continuing to load more content can continue until no further additional content is loaded in the document. The process of scrolling the document can additionally or alternatively continue until a threshold is reached. A “show more” control may be identified and activated after each scroll action. For example, a threshold amount of content in the page may be reached, or a predetermined threshold number of cycles of scrolling and activating a “show more” button may be reached, e.g., 5 cycles, 10 cycles, etc.

At step (G), the computing device may process the additional content 166 to update a document index 170. For example, the document index is updated by associating the additional content with an identifier for the electronic document in the document index. That is, the document index includes a table storing at least associations between identifiers and content associated with the identifiers. For example, the computing device can associate the additional content with an identifier that identifies the electronic document. When the electronic document is identified by the identifier for further processing, e.g., for indexing of content for search results, the identifier can identify the additional content of the electronic document (as well as the initial content of the electronic document).

At step (H), the computing device may process the additional content to generate a transcoded version of the electronic document. For example, the transcoded electronic document 180 can be a transcoded version of the electronic document 160. In this example, generating the transcoded electronic document by the computing device can include transcoding the (dynamic) electronic document 160 to a static HTML version of the electronic document 160 based on the additional content. The transcoded electronic document is a static HTML version of the electronic document 160. Thus, when a client computing device having a slow network connection, or an intermittent network connection, is accessing the electronic document, the transcoded electronic document can be provided (e.g., by the computing device). This enables efficient user of bandwidth and computing resources of the client computing device and the computing device serving the transcoded electronic document.

Example electronic document

FIG. 2 shows an example electronic document 200 similar to the electronic documents 120a, 120b, or 160. The electronic document can be displayed in a web connected browser provided for display by a client computing device. For purposes of indexing, transcoding, crawling, etc., the document may be loaded by a server system for automated processing without user viewing or interaction.

The electronic document can include content 202 that is loaded to provide an initial view of the document. In this example, the content includes a plurality of search results. The electronic document includes a number of interactive UI elements 210, 212, 214, 216. However, not all of these elements would be effective for loading additional content into the document. Of these, element 216 represents an element that will expand the loaded content of the page by loading additional page content.

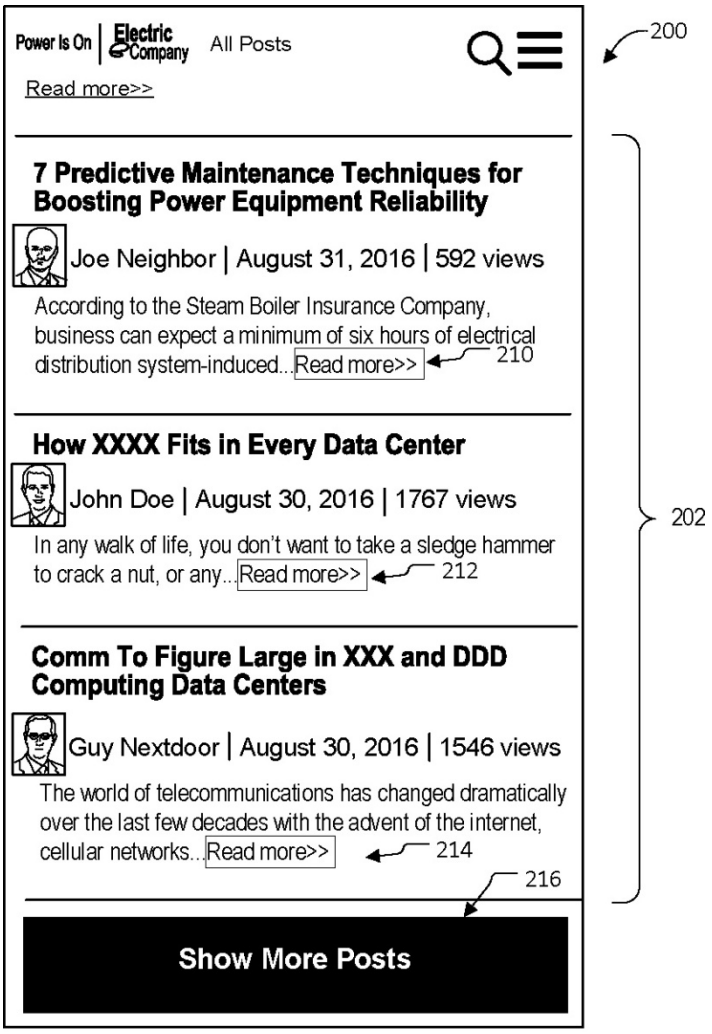

FIG. 2

When crawling or searching the document and others like it, the computing system can quickly identify the "show more" element 216 and distinguish it from other interactive UI elements using the templates noted above. These templates, being validated for specific URL patterns, can direct the computing system to the location and identity of the "show more" elements in documents without requiring an analysis of the content of each document as it is loaded. This also helps the computing system avoid activating elements such as the UI elements 210, 212, 214 which would not initiate loading of additional content from the server. For example, the UI elements 210, 212, 214 may represent elements that would navigate away

from the current document, and thus activating them would be counter to the goal of obtaining the rest of the document. Or, the UI elements 210, 212, 214 may trigger display of content that was visually hidden, but already served and present based on the initial loading, so the elements would not obtain additional content of the page. As noted above, the automatic activation of the "show more" control is to trigger the acquisition of further content for the page, and hiding or displaying content already obtained does not achieve this.

A computer device of various types may be used with the techniques described here, e.g., laptops, desktops, workstations, personal digital assistants, servers, blade servers, mainframes, and other appropriate computers. Various forms of mobile devices may also be used, such as personal digital assistants, cellular telephones, smartphones, and other similar computing devices. Multiple computing devices may be connected, with each device providing portions of the necessary operations (e.g., as a server bank, a group of blade servers, or a multi-processor system).

CONCLUSION

This disclosure describes techniques for a computer system to automatically initiate loading of and obtaining additional document content that is not served or transferred as part of the initial loading of a document. Features allow efficient indexing of content of web pages by loading much more of the content of a document than is available using typical crawling techniques. Documents that are served in limited increments can be obtained and indexed more completely. By indexing this additional content and making it available to a search engine, the search engine can provide more accurate results. For example, search results can be generated based on the entire contents of document, rather than based on only a limited portion initially

provided for a partial view of the document. The system may avoid multiple network round trips and interactions between the client device and a search server as the user seeks more accurate search results upon dissatisfaction with initial search results.

Additionally, features allow efficient serving of documents to devices connected to networks having high latency and/or low bandwidth. For devices that have high latencies for network communication, documents that require repeated loading of content can repeatedly incur the high latency of the connection. However, a server system configured to obtain the additional content can transcode dynamic documents requiring iterative loading and generate a transcoded version of the document, e.g., a static HTML version of a web page as opposed to a dynamic web page. The static HTML version of the web page can be provided (e.g., served) to client devices in network environments experiencing high latency and/or low bandwidth to provide efficient utilization of network resources and computing resources of the client device. As a result, a device loading the transcoded version can avoid the overhead and delays that would otherwise be incurred by sequentially loading a document in stages.