Implementing a Web Annotation System for Supporting Cooperative Works Using Tablet Devices

Hiroyuki Sano  Shun Shiramatsu  Tadachika Ozono  Toramatsu Shintani
Dept. of Computer Science and Engineering
Graduate School of Engineering, Nagoya Institute of Technology

Abstract

To support cooperative works by using tablet devices, a web annotation system and a smart sinage system are implemented in this work. We propose a web annotation system which adds the functionality of stickies to web pages, adds comments to web pages, and creates bidirectional links between the stickies. We also propose a smart sinage system for tablet devices to support collaborative works based on Web contents. The smart sinage system can synchronize displayed contents and annotation stickies on the system. In this paper, we describe the outline of these two systems and implementation of the systems. Two experiments were conducted to show that these proposed systems have potential in practical use.

1. Introduction

We implemented a web annotation system which adds the functionality of stickies to web pages and creates bidirectional links between the stickies. To apply the annotation system to cooperative works on tablet devices, we also implemented a smart signage system, which can synchronize displayed web contents and annotations on the system.

The web annotation system proposed in this paper automatically generates bidirectional links between annotation stickies referencing similar information and subsequently categorizes them. Our system has the functionality of stickies to web pages. The stickies allow for important parts of a web page which contains large amounts of data to be highlighted. Such stickies and links can be used as user preferences, and have the potential to become a much better alternative to bookmarks and tags.

Users can annotate web contents by providing comments, tags and links inside stickies. Web pages contain texts, images, and other types of information which are often related to more than one topic. By using our system, users can point out specific contents more accurately than by using bookmarks in web browsers or current tagging systems, and users also make bidirectional links between stickies.

These stickies can be shared in realtime between some users by using our smart sinage system proposed in this paper. The smart sinage system has a function to synchronize displayed web contents and annotation stickies on multiple tablet devices. This function is useful when users have cooperative works on their tablet devices because all devices mirror their screens by our synchronization module. Users can discuss about web contents displayed on their tablet devices via web annotation stickies.

The rest of the paper is organized as follows. Section 2 reviews related works and systems in the area of Web annotation and computer supported collaborative works based on tablet devices. In section 3, we propose a web annotation system that eables users to place stickies on web contents and to provide comments in relation to the content referenced by the stickies. In section 4, we propose a smart sinage which can synchronize displayed Web contents on multiple tablet devices. The system can support users' cooperative works via web annotation stickies. Experiments and their results are reported in section 5. Finally, we conclude the paper in section 6.

2. Related Works

First, we introduce several studies about web annotations. There are several annotation systems for web pages. Annotea[8, 9], which was developed by W3C, is a framework which allows annotations to be placed on web pages. Annotea is a Semantic Web based project for which the inspiration came from users' collaboration problems in the Web. It examined what users did naturally and selected familiar metaphors for supporting better collaboration. Annotea is available for Firefox through the Annotea Ubimarks extension, as well as for Amaya, which is an open-source web browser developed by W3C. However, although Annotea enables users to create links from one web page to other web pages, this needs to be done manually.
There also exist social tagging systems, where users add tags to web contents and share those tags with other users[13]. Chirita, et al. propose P-TAG, a method which automatically generates personalized tags for Web pages. Upon browsing a Web page, P-TAG produces keywords relevant both to its textual content, but also to the data residing on the surfer's Desktop, thus expressing a personalized viewpoint[3]. However, to express relations between tags is difficult, although tags are useful for specifying both web pages and web contents.

Some studies show that annotations can be useful for education systems. For example, D. Giordano and S. Mineo propose a graphical annotation system for a distributed e-learning architecture. The role of annotations for e-learning is discussed and technologies for web annotation are reviewed[6]. Farzan and Brusilovsky presents their attempt to integrate annotation and adaptive navigation support for open corpusweb educational resources into a single value-added service. The AnnotatEd system offers both annotation (through highlighting and free-text comments) and adaptive navigation support (through social navigation). It works in the traditional intermediary way, by standing between a web page and its user[4].

Next, we introduce several different display systems already in place to support users' collaborative works by using tablet or handheld devices. Cheng et al. present a system that supports the use of tablet devices for interaction and collaboration with large displays in the paper [2]. Users can interact with a subset of the large workspace on their tablet, while the same area is visualized on the large display as a rectangular frame. In the paper [10], Liu et al. propose a shared display groupware and explores whether the use of shared displays in classrooms can augment the handheld devices and enhance the effectiveness of handheld devices in promoting communication among participants.

Our system also aims to support collaborative works on users tablet devices. Although the systems proposed in the paper [2, 10] require a large display to share the information, our system requires no large display because displayed contents on users' tablet devices are synchronized for sharing the information and users can share web annotations on the system.

3. Web Annotation Stickies
3.1. Outline of the system
The web annotation system enables users to place stickies on web contents and to provide comments in relation to the content referenced by the stickies. Users can place annotation stickies on all types of contents on web pages, including text data, images, and so on, by using the web annotation system.

In the system, a web agent, which is referred to as a “biLink agent”, keeps track of the stickies which users have placed on web pages. The biLink agent is constructed from a page agent and a base agent, using the web agent model “MiSpider[5]”. The page agent sends to the base agent the web content on which a user has placed a sticky. Then the base agent classifies the stickies by using information which it has received from the page agent, and generates bidirectional links between the stickies placed on similar contents.

Figure 1 shows the outline of the system. The system comprises a “Fusen client” and a “Fusen server” 1. The Fusen client, which is written in JavaScript, runs on the user's web browser and acts as a page agent. The Fusen client provides an interface for placing stickies on web contents. The Fusen server is the system which saves the stickies which users have placed. The Fusen server is a proxy server, which acts as a base agent, and a database for saving the properties of the stickies.

When a user requests a web page from the web annotation system, their web browser obtains the HTML source code of the web page from the proxy server, which contains the database with the properties of the stickies. The proxy server accesses the database and sends an inquiry regarding whether the HTML

---

1 Fusen means “paper annotation sticky” in Japanese.
source code of the web page has been saved to the database. If the page has been saved, the proxy server sends the HTML code to the web browser. If not, the proxy server obtains the code from the relevant web server, adds a Fusen client to the HTML code, and sends the HTML code thus generated to the web browser. There are two methods for adding web services to an existing web page: one involves a proxy server[11], and the other is based on bookmarklets[12]. We adopted the proxy server method for the system. If a user places a new sticky or updates an existing sticky, the Fusen client updates the database on the Fusen server when the user leaves the web page.

Figure 2 shows a screenshot of a web annotation sticky provided by our system. In this example, a web browser for PC is used, but users can use our annotation system via web browsers for tablet devices. Users can place annotation stickies on web contents as shown in Figure 2. When a user double-clicks an annotation sticky, they can see detailed information about the sticky in a dedicated popup window, which can be closed by double-clicking on the sticky again. In the example, the user has placed two annotation stickies on the page and has opened the popup window of the lower sticky. Regarding points (1) to (5), which are shown in the popup window in Figure 2, (1) indicates the date when the user placed the sticky, (2) is used when the user wishes to change the color of the sticky, (3) is a comment to the referenced content (this comment is also displayed on the sticky image), (4) shows links to similar contents (when a user clicks on a link, they can see the stickies which have been placed on similar contents), and (5) is used for deleting the sticky.

3.2. Deciding the place of a sticky using a DOM tree

If an annotation sticky is displayed based on absolute coordinates, the sticky will not move when a window size or a font size of a web browser is changed in spite of the fact that the absolute coordinates of the content might change. As a result, the position of the sticky ceases to match the position of the content which the sticky refers to (e.g., Internote::Firefox Add-ons\(^2\)). In order to avoid the problem, our system appends a HTML <img> tag to show a sticky image to the DOM node clicked by the user. By using the DOM-based method, both the sticky and the content it refers to are displayed at the same position even if the absolute coordinates of the content changes.

The method produces a new problem related to the fact that when users wish to place a sticky inside a very long text, the sticky does not appear at the desired place. In the system, if a DOM node where the user is attempting to place a sticky is a text node, the system divides the node into span nodes. If a text node is divided into span nodes that have only one character, the place where users can place a sticky can be chosen with an accuracy of one character, which enables users to place stickies practically anywhere on a web page. However, dividing text into span nodes of one character can create a new problem where the beginning of a new line might change drastically when compared with the unmodified web page. This is due to the differences in the implementation of rendering engines in web browsers. In our system, when users place a sticky inside a text, the system divides the text node into span nodes containing only one word. As the orthography of Western languages, such as English or French, demands a space to be left between words, the system can easily recognize words in such languages. Unfortunately, the orthography of Japanese does not leave a space between words, and therefore the system cannot easily recognize Japanese words. For this reason, the system needs to be able to recognize morphological units as words. The system performs a morphological analysis of a text node and divides the node into morphological span nodes. A page agent sends text on which the user has clicked to the Fusen server, which analyzes the text morphologically by using a Japanese Morphological Analyzer (MeCab\(^3\)) and adds <span> tags between morphological units. The base agent then sends the divided nodes back to the page agent. The division of the nodes based on morphological units provides sufficient precision for placing stickies in the system.

3.3. Bidirectional links between stickies

Weblogs implement a function called the “trackback”, which informs weblog authors about what kinds of web pages are linking to articles in the weblog. In this sense, the trackback feature makes weblogs bidirectional. In our system, a biLink agent generates links between stickies with a similar content based on the concept of

\(^2\) https://addons.mozilla.org/firefox/addon/2011

\(^3\) http://mecab.sourceforge.net/
trackback, and as a result users can traverse the stickies by using those links. We refer to the agent as “biLink agent” and the links as “bidirectional links”.

A biLink agent implements a function for automatic generation of bidirectional links between stickies. A biLink agent constitutes a page agent and a base agent, which are used for keeping track of the stickies placed by the user. When a user places a sticky, the page agent extracts the text around the content where the sticky is placed by looking at the DOM tree. This process is based on heuristics, in other words, on the text around image files or flash files which describes those files[1]. The extracted text is sent to the base agent, which analyzes the text and classifies the sticky in accordance with the classification method described below. Subsequently, the base agent automatically generates bidirectional links between stickies placed on similar content.

Yang et al. examined some approaches to classify hypertext documents[14]. Glover et al. analyze the relative utility of document text, and the text in citing documents near the citation, for classification and description[7]. We present the method which the biLink agent uses to classify the stickies. The biLink agent uses MeCab to parse the web page containing the information on which the user has placed a sticky and decides the index terms of the web page. The biLink agent then uses the values of term frequency-inverse document frequency (TF-IDF) as evaluations of the index terms of the web page. The base document whose TF value is calculated by the biLink agent is the web page containing the information which the user has referenced with a sticky. Since the system is a web-based application, the biLink agent uses the total number of web pages which the Yahoo! API can search as the total number of documents, and the number of results which the Yahoo! API obtains appears as the number of index terms when the agent calculates the IDF.

The similarity between documents in classifying stickies is calculated by using a cosine measure based on the Vector Space Model. Each dimension of a document vector corresponds to a separate term, and each component corresponds to an evaluation of the term. However, the biLink agent performs the calculation by assigning a certain weight to the content which is referenced with a sticky. The term “content” here indicates the nearest block-level element, where the tracing is in the direction from the node where the sticky is placed toward parent nodes.

Document vectors in the system are calculated by the following formula.

$$d_i = (w_{i1}, w_{i2}, ..., w_{iM})^T + \alpha (v_{i1}, v_{i2}, ..., v_{iM})^T$$

The similarity between documents in classifying stickies is calculated by using a cosine measure based on the Vector Space Model. Each dimension of a document vector corresponds to a separate term, and each component corresponds to an evaluation of the term. However, the biLink agent performs the calculation by assigning a certain weight to the content which is referenced with a sticky. The term “content” here indicates the nearest block-level element, where the tracing is in the direction from the node where the sticky is placed toward parent nodes.

Document vectors in the system are calculated by the following formula.

$$d_i = (w_{i1}, w_{i2}, ..., w_{iM})^T + \alpha (v_{i1}, v_{i2}, ..., v_{iM})^T$$
cluster which contains only the first sticky. After that, as more stickies are placed on parts of the page with different contents, the biLink agent analyzes the content in those parts, and calculates the document vectors as well as the similarity between the document vectors and the document vectors of the clusters which already exist. If the similarity is greater than a predefined threshold, the biLink agent adds the sticky to the cluster and updates the document vector of the cluster with the average vector. If the similarity is lower than the threshold, the biLink agent generates a new cluster and adds the sticky to the new cluster.

The weight $\alpha$ and the similarity threshold are decided on the basis of the results performed by a person. A certain number of web pages are collected at random and classified manually, after which the agent also classifies them. In order to match the results of the manual classification with the results of the classification performed by the agent, we adjusted $\alpha$ and the threshold. Eventually, the most satisfactory level of conformance was attained when $\alpha$ was 25 and the threshold was 0.15.

If a sticky is classified into an existent cluster, the biLink agent generates bidirectional links between the sticky and all other stickies in the same cluster, and the final result is that the bidirectional links form a complete graph.

4. Synchronizing Displayed Web Contents and Annotation Stickies
4.1. Smart sinage system

We implemented a system which can synchronize displayed Web contents on multiple tablet devices. We call the system “Smart Sinage System” and contents for the smart sinage system are called “sinage cards” in this paper. Sinage card is Web contents written in HTML and JavaScripts. The web annotation stickies, which are provided by the system described in section 3, can be placed on sinage cards because signage cards are Web contents. The web annotation stickies and smart sinage system will be useful for supporting cooperative work. Users can make annotations to sinage cards via web annotation stickies and the stickies are displayed on all devices in synchronization with that are used in smart sinage system. Users can discuss the displayed cards on their tablet devices by using the annotation stickies.

Figure 4 shows an example of synchronization. The tablet devices in the proposed system have two modes, one is an operator mode and another is a viewer mode. A displayed content on a device which is set to a viewer mode is synchronized with that on a device which is set to an operator mode. A device in an operator mode sends synchronous messages to other devices. And the devices that received the synchronous messages redraw their displayed contents based on the messages. In Figure 4, user 1 uses a device which is set to an operator mode, and user 2 and 3 use devices which are set to a viewer mode. The devices user 1 and user 2 are using are now displaying sinage card “B”, while the device user 3 is using is now displaying sinage card “C” in this example. When user 1 changes the displayed card to card “A” on the device, the device sends a synchronous message to the devices user 2 and user 3 are using. The devices user 2 and user 3 are using redraw their dispayed cards based on the message, and those devices also displays card “A”. So, the same cards are always displayed on these three devices.

4.2. Sending a synchronous message

Here, we discuss an implementation of the synchronization module of the smart sinage system. The synchronization module checks user’s interaction. A device that is set to an operator mode sends a synchronous message to other devices. Synchronous messages must be send all devices in the smart sinage system. A method to manage IP addresses of all devices in an integrated fashion is a cumbersome procedure because the IP addresses table must be updated every time when devices increase or decrease. To attack the problem, UDP broadcast is adopted in the system to send synchronous messages. Broadcast is sending packets to all nodes in a local area network. An operator send a synchronous message to broadcast
Table 1: Structure of the synchronous message

<table>
<thead>
<tr>
<th>attribute</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) ip_addr</td>
<td>IP address of the sender, which is the machine that sent the message</td>
</tr>
<tr>
<td>(2) date</td>
<td>UNIX time when the message was sent</td>
</tr>
<tr>
<td>(3) url</td>
<td>URL of the displayed content</td>
</tr>
<tr>
<td>(4) scale</td>
<td>Zoom level of the displayed content</td>
</tr>
<tr>
<td>(5) (scrl_x, scrl_y)</td>
<td>x and y coordinates of the scroll offset</td>
</tr>
</tbody>
</table>

address and all devices receive the message and refresh their screen based on the message.

The synchronous message contains five attributes listed in Table 1. We will describe these attributes in detail.

(1) ip_addr is the operator’s IP address. ip_addr is used not to process based on the synchronous messages from own device. UDP broadcast message reaches to the device that the message sent. It is not needed to synchronize based on own synchronous messages. Devices that received the message determine if the message was sent from other devices or not by using the ip_addr attribute.

(2) date is the time when the message was sent by the operator. The date attribute is in UNIX time. date is used to synchronize based on the newest message. UDP header does not have a sequence number. So, when two messages are sent by UDP, the second message will reach before the first one will reach. UDP does not promise rt(m) < rt(m+1), often causes rt(m) > rt(m+1).

(3) url, (4) scale and (5) (scrl_x, scrl_y) mean the state of displayed contents on the operator. (3) url is the content URL that is displayed on the operator. (4) scale is zoom level of the displayed content on the operator. And finally, (5) (scrl_x, scrl_y) are x and y coordinates of the scroll offset on the operator.

The system repeats sending messages to avoid packet loss. We conduct a preliminary experiment to determine the number of times messages to send, and the results show that messages should be sent in five repetitions.

4.3. Algorithm for synchronization

Figure 5 shows the procedure receiveSyncMessage that is called when a synchronous message has reached to a device. The first line in Figure 5 determines whether the arrived message was sent from self device or not, and the message is the newest message or not. The last_sync_date must be allocated as a global variable while the system running. When the message was sent from other devices and it was the newest message, the process for synchronization (after the third line in Figure 5) begins. The lines three to five in Figure 5 are the process to show the same contents on all devices. If a URL of a content that the device is displaying is different from displayed on the operator, a contents will change to the one that is displayed on the operator. The lines six to eight in Figure 5 are the process for zoom scale synchronization. If the zoom scale of the device is different from the operator, the zoom scale of the operator will set to the device. The lines nine to thirteen in Figure 5 are the process for scroll offsets synchronization.

5. Experimental Results

5.1. Speed of placing stickies

We had an experiment to show that the web annotation system proposed in section 3 can make stickies at the time of practical use. We measured the change in the processing time in relation to the increase of the text length. More specifically, we evaluated amounts of text data in the range of 200 bytes to 2,000 bytes, where the step of increase was at the rate of 200 bytes. We placed stickies 10 times on each text and measured the time from when the mouse button was clicked to when the web browser received the results of the analyzed morphological parts from the Fusen server. We also measured the time from when the mouse button was clicked to when the sticky was displayed.
In order to avoid the influence of inherent network delays, the base agent and the page agent were executed on the same computer. In other words, the Fusen server and the web browser were running on the same computer. The relevant specifications of the computer system used in the experiment are outlined below. The experiment was performed on the MacBook Pro (Early 2006) that has Intel Core Duo 1.83GHz CPU and 1.5GB DDR SDRAM. The Operating System running on the machine was Mac OS X 10.5.3 and the Web browser was Safari 3.1.1.

Figure 6 shows the processing time for the morphological analysis and for displaying the sticky. The horizontal axis shows the text length in bytes, while the vertical axis shows the processing time. The graph plotted with circles (upper graph in Figure 6) is the processing time needed for displaying the sticky, and the graph plotted with squares (lower graph in Figure 6) is the processing time of the client receiving the results of the morphological analysis from the server. Figure 6 shows that the longer the text length, the longer the processing time. However, the fact that the system processes the information in 170 ms when the text length is 2,000 bytes is strong proof that the method is very fast. Since the content used in the system is part of a web page, a text length of 2,000 bytes is sufficiently long for practical purposes.

Thus, the experiment shows that stickies can be placed on the page very quickly, and that the proposed method has a potential for practical use.

5.2. Speed of synchronization

We conducted another experiment to show that the smart sinage system proposed in section 4 can synchronize displayed contents at the time of practical use. We measure the processing speed of the proposed synchronous method.

Figure 7 shows the processing time for changing the displayed content and for scrolling the content, and for zooming the content. We runned the system on one iOS simulator and multiple iPad devices. The system runned on iOS simulator was set to operator mode, and runned on iPads were set to viewer mode. The results show that the system can synchronize within a given length of time without the number of the devices that are used in the experiment. This means that UDP broadcasting is suitable for sending synchronous messages. The results show that the system can process in approximately 1,000 milliseconds for scroll and zoom synchronization. The time for changing the displayed content are over 3,000 milliseconds and the getting HTML of a new signage card from web server via http protocol is considered a major cause of these overheads.

Thus, the experiment shows that the system can synchronize a displayed content very quickly, and that the proposed method has a potential for practical use.

6. Conclusion

A web annotation system and a smart sinage system are implemented in this work to support cooperative works by using tablet devices. The proposed web annotation system enables users to add the functionality of stickies to web pages, add comments to web pages, and create bidirectional links between the stickies. We also propose the smart sinage system for tablet devices to support collaborative works based on Web contents. The smart sinage system can synchronize displayed contents and annotation stickies on the system. We described the outline of these two systems and
implementation of the systems in this paper. The experiments showed that stickies can be placed on the page very quickly, and the system can synchronize a displayed content within a given length of time without the number of the devices that are used in the experiment.

Acknowledgments
This work was supported by Hori Sciences & Arts Foundation in Japan.

References