A New Annotation System for Dynamic Web Pages Driven by NLP

Mohammad Smadi, Mohammad Hawash, Omar Daqqa, Amjad Hawash*, Ahmed Awad*
Information & Computer Science
An-Najah N. University
Nablus, Palestine
mo.smadi@outlook.com, mohammadahawash@gmail.com, omardaqqa2018@gmail.com, ahmedawad@najah.edu, amjad@najah.edu

***Abstract*—Web annotation has become an essential technique to express people's thoughts and feelings by attaching annotations to web content. Annotators with the same interests can exchange their ideas and experiences by conducting universal online collaborations. The idea of submitting annotations depends on attaching vocal or textual notes with the contents of websites. However, annotating dynamic data is a problem that encourages researchers to work on proper solutions. Losing annotations because of the change in website annotated contents will definitely lead to losing the intended collaboration between annotators. This work is related to annotating dynamic websites by computing the textual similarity between erased (or relocated) annotated text and the remained text in dynamic websites by exploiting NLP (Natural Language Processing) algorithms. The attached annotation with dynamic content will be attached to the most related text on the website. By this, annotators will not lose their annotations and replies and hence their collaboration will remain. The experimental tests conducted in this work reflect promising results.**

***Index Terms*—Web Annotation, NLP, Dynamic Webpages.**

I. Introduction

Web browsing plays an important role these days in finding proper information. The existence of digital documents with the facilities of web services enables users to find the information they seek easily and accurately. However, searching for proper information in the tremendous amounts of websites consumes both time and effort [1] [2]. Web annotation (by adding illustrative notes attached to different types of web content) improves the value of information and makes it easier for knowledge seekers in finding their needed information as well as enhancing the online collaboration between users [3].

Recently, the structure of annotations is not limited to textual ones, vocal and drawings are also famous structures of annotations. The variety in structure is very important because annotators have different ways to express their feelings whilst the textual annotations only limit expressing these feelings [3] [4]. Creating annotations by drawing shapes over the contents of websites also became a famous technique. Using this type of annotation, users are able to draw shapes over some contents of the websites and attach these drawings with textual notes. These drawings are shared between users of such tools to conduct some kind of discussion based on the drawings and their notes [5]. Several tools exist on the web to enhance this kind of annotation. Sketchpad (https://sketch.io/sketchpad/) , Sketch to Web (https://sketch-to-web.com/) and MADCOW are all examples of famous tools related to this kind of annotations [6]. Vocal annotations are also famous depending on the fact that the tones of voices have information by themselves. Sometimes expressing feelings and the intention of some conversation becomes easier in the case of vocal notes and abbreviates several amounts of textual data [3].

Corresponding Author: amjad@najah.edu, ahmedawad@najah.edu

The continuous improvement of website contents is necessary in order to gain more visitors and keep the websites more attractive. Despite the usefulness of annotating websites and their role in increasing the annotators' collaboration through the existence of various annotations' structures, this field suffers from the continuous change of annotated contents either by removing these contents or by changing their locations on the websites they exist. Dynamic contents of the web are related to the continuous change in the contents of websites like magazines, newsletters, and blogs. Annotations linked to the contents of such websites are prone to be lost since the link between the annotation and the content becomes broken in case the content is changed or moved to other sections of the website. Consequently, this leads to two problems:

1) Losing annotations: when annotators (users) submit annotations for some website content, the annotation system creates a link between the annotation itself and the content taking into account the position of the content on the website, usually saved in a dedicated database. When the content is removed or moved to other sections, the link between it and its related annotation(s) is lost as well due to the change of the content position. In this case, although the annotation itself is still saved in the database, the annotation system will not be able to attach the annotation(s) with the annotated content anymore [7].
2) Decreasing users' collaboration: one of the aims of annotation systems is to increase users' collaboration. In the case users are unable to retrieve their annotations involved

in some online conversation, the intended collaboration will decrease and hence losing users' interest [8].

In this work, we propose a solution to the problem of dynamic contents of websites represented in searching for the most related text to the annotated one. In the proposed framework, the developed annotations prior to website changes are attached to the most related text that appears in the updated website with the highest textual similarity following an NLP algorithm. In this way, the collaboration between users is preserved in dynamic websites. Our contributions to this work are summarized as follows:

- We develop a simple textual-based annotation system architecture in order to present the problem of annotating the dynamic content of the websites.
- We utilize an NLP algorithm to find the most related text in a website to the removed annotated text in order to link the annotations to the searched one.
- We conduct a complexity analysis for the proposed system in terms of runtime.
- We conduct an experimental test in order to measure the amount of raised collaboration between annotators of dynamic contents of websites.

The rest of this paper is organized as follows: Related work is presented in Section II. The problem statement is proposed in Section III whilst the proposed system architecture is discussed in detail in Section IV. Then, we discuss complexity analysis in Section V. Experimental tests are explained in Section VI, whereas, Section VII concludes this paper.

II. Related Works

Several annotation tools exist and are used extensively by universal annotators. These tools provide several services in order to enhance the process of submitting annotations. Darwin V7 (https://www.v7labs.com/), Deepen (https://www.deepen.ai/) and Heartex (https://heartex.com/) are all annotation tools used for computer vision. ClickUp (https://clickup.com/), Filestage (https://filestage.io/), Prodigy (https://prodi.gy/) and Annotate (https://www.annotate.com/) are all up to date online annotation systems. Part of these annotation tools are related to annotating online audio and video files, others are used to annotate online PDF files, and others are used to annotate textual contents of websites. Despite the variety of usage of these annotation tools and others, they lack the property of retrieving annotation for removed content. Users of such tools suffer from losing their annotations and replies because the related contents (text, audio, video) are removed or moved to other locations on the websites. However, annotating dynamic websites is a problem in terms of annotation retrieval. Continuously changed websites in terms of their contents is a problem for web annotation because the relations between the highlighted texts and their corresponding annotations are lost due to text changes [9].

Literature contains several workarounds to tackle the problem of annotating dynamic web pages. Semantic annotation is one of these techniques that is related to annotating text according to its semantic and not textual content. The work presented in [10] is related to examining semantic annotation systems in which the study concluded that the annotators have to gain enough knowledge of the contents they are annotating and have to understand exactly the intended semantic behind these contents. However, not all users gain this experience and this led to annotating wrong materials. Moreover, the work of [11] is related to list some annotation tools with the languages used to extract data from them.

The work published in [12] addresses the task of annotating web pages whose dynamic content is derived from a database. The approach adopted is based on annotating a database schema based on public ontologies and using this database annotation to generate a dynamic web page's content annotation on the fly. Despite the work being related to annotating dynamic contents retrieved from the database with their relationships as entities, the work did not touch on the problem of losing annotations due to the update of textual content of the annotated materials nor the problem generated by moving the same content to other website locations.

Authors of the work published in [13] worked on the automatic generation of a function that dynamically calculates annotations for a Web document and they introduced a novel approach for annotating dynamic web documents by annotating the queries enveloped in each document and their results. Again, the work is related to annotating queries of documents and not directly related to the dynamic contents of these documents.

Our contribution in this work goes directly to the main problem of annotating dynamically changed contents and answers the question: *"How to retrieve annotations linked with dynamic contents?"*, that is how to solve the problem of broken links between annotations and their related web contents. We work on computing the similarity between the annotated content (that does not belong to the website anymore) and the rest of the contents on the main website. In case the similarity between the deleted annotated content and any other content of the same website is acceptable, then the annotations are linked to the most similar content. In this case, the possibility of losing annotations will be decreased and thus, the amount of collaboration between annotators will be positively affected. The implemented algorithm in the backend server of the proposed architecture uses NLP techniques to compare the annotated texts that retrieved from the database with the contents of the visited website. Upon some match, the system highlights the matched texts and displays their corresponding annotations.

III. Problem Statement & Proposed Methodology

This work is related to proposing a solution for the problem of annotating continuously changed contests of websites. The amount of intended collaboration between annotators is decreased when the annotated contents of websites are removed or changed in their textual data or even moved to other sections of the same website. The successful retrieval of annotations for a given website and displaying them near the texts where they were linked during their creation depends mainly on the spatial links of these annotations with the corresponding texts. Any change in the textual data, removing or changing their positions will break these links and as a consequence, the annotation system will not be able to link corresponding annotations and this, in turn, decreases the amount of intended collaboration between annotators.

To tackle the aforementioned problem, we propose a comprehensive framework driven by the NLP algorithm to link the existing annotations for a given web page to the most similar context in the modified version of that web page. The proposed framework is presented as a tool composed of subsystems whose architecture will be explained in detail, subsequently.

IV. System Architecture

Submitting annotations for some websites is considered as adding a transparent layer on top of the website, containing a set of links between annotations and their related contents. Visitors of an annotated website are able to view annotations in that layer with the possibility of submitting new annotations and/or replies for the annotations previously submitted by others. In all cases, the annotation system creates links between the annotations that exist in the transparent layer and the contents of the websites being annotated. These links accompanied by the other annotations data are all saved in a dedicated database. The simple textual annotation system we develop for this work contains these functionalities as depicted in Figure 1 below where the major components of the implemented annotation system exist.

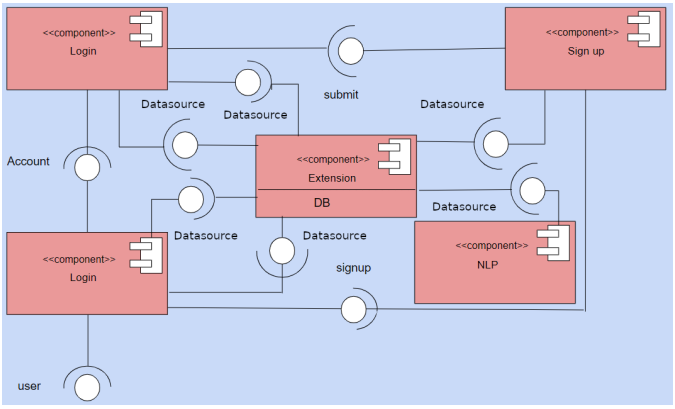

Fig. 1. System Components.

Figure 2 depicts the client/server architecture of the proposed framework wherein the major components are shown. The client-side (front-end) of the system is composed of JavaScript code to implement the Google extension that forms the interaction point between the annotators and the system.

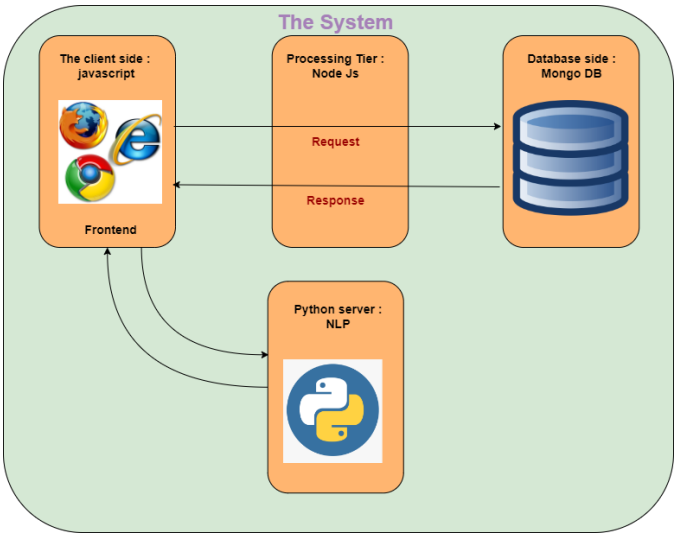

Fig. 2. Proposed System Architecture.

The server side of the system is composed of *Node JS*, *Python code* and *Database server*. These parts collaborate with others to implement the simple annotation system we developed as well as implementing our proposed solution for annotating the dynamic contents of websites. The different parts of the system are deeply discussed in the following subsections.

A. Google Extension

We implemented a *Google Extension* in order to have an interaction point between users and the system. The JavaScript and the *ReactJS* codes embedded in the *Google Extension* contacts both *Node JS* and *Python code* in order to (1) save submitted annotations with their related data, (2) retrieve previously submitted annotations for a given user and website, and (3) execute necessary processing to work around the annotation of dynamic contents of websites. Figure 3 depicts the simple structure of the database that composed of two main entities: *Annotation* and *User*. The *Annotation* entity is used to save annotations with their related data such as the highlighted text, the annotation text, the URL of the website, and the placeholder used to save the position of the highlighted text of the annotation within the URL. The reflexive relationship of the *Annotation* entity is to depict the relationship between the annotations and their replies in which each annotation could have 0 or more replies. The relationship between *User* and *Annotation* is one-to-many since a user could have 0 or more annotations. The *User* entity is used to save data about the users of the system.

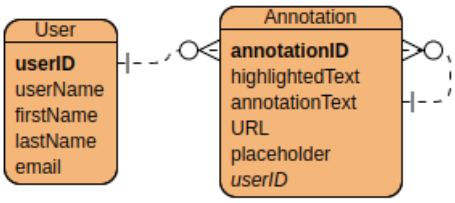

Fig. 3. Entity Relationship Diagram.

Google Extension services can be summarized in:

1) Website Content Highlighting: It begins when a user visits a website and highlights some interesting textual content with the mouse. According to this and upon clicking the right mouse button, the command *"Click to add a comment"* appears within a list of commands in a side window. When this command is clicked, a pop-up window appears with a set of controls to enable the users to create their annotations. Figure 4 depicts this facility.

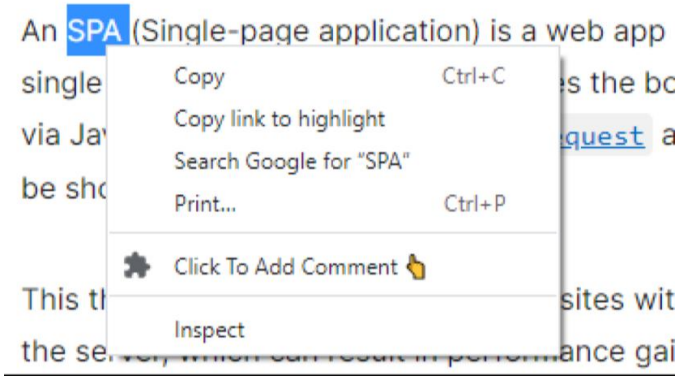

Fig. 4. Highlighting some text.

2) Annotation Creation: Upon the appearance of the pop-up window and when the user finishes writing his/her comment and clicks *"Submit Comment"*, the *Google Extension* interacts with the *Node JS* to deliver the created annotation along with its related data. Figure 5 depicts the annotation submission process.

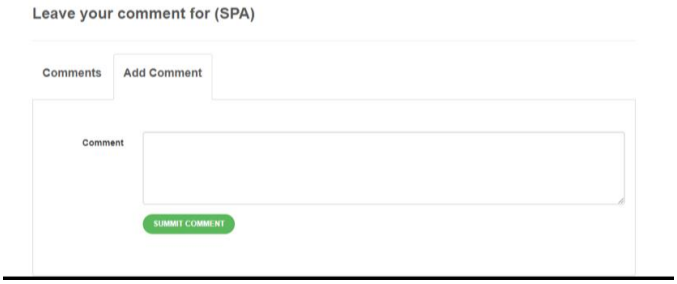

Fig. 5. Creating new annotation.

Figure 6 illustrates the process of annotation creation and submission. The activity begins with the highlight of some textual content by some user for a given website represented in the function *highlight(text)* identified in the object *Google Extension* that executes the function *display(text)* identified in the object *Pop-Up Window*. Upon the invocation of the function *display(text)*, the object *Pop-Up Window* notifies the user with the function *notifyUser()* in order to fill up the needed data to complete the process of annotation creation. The user then executes the function *create(annotation)* on the object *Pop- Up Window* that executes the function *submit(annotation)* identified in the object *Node JS*. The object *Node JS* then executes the function *save(annotation)* on the object *Mongo DVMS* that saved the annotation in the system database. Upon the successful save of the annotation, a sequence of replies is executed which is represented in *save=true* on the objects *Node JS* and *Pop-Up Window* and the call-back function *notifyUser()* is executed to finally notify the user with the successful annotation submission.

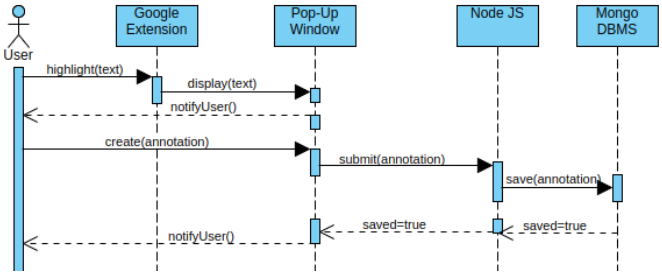

Fig. 6. Creating new annotation Sequence Diagram.

3) Displaying Annotations: When users visit some website, they are able to ask the *Google Extension* to load all annotations related to that website, either submitted by themselves or by other users. As a result, the *Google Extension* asks the database through the *Node JS* to load all related annotations relying on the URL of the current website as a key and executes the *Tree Walker* algorithm in the *Python Code* component to search for the texts of annotations inside the website regardless of its modification and/or a change in their locations. Upon the match of the annotated texts and the contents of the website, the extension highlights the contents and when the users hover the mouse above some highlighted text, the extension displays a list of pop-ups containing all of the annotations that were previously created by the user or other users. *TreeWalker* (https://docs.rs/web-sys/0.3.44/web_sys/struct.TreeWalker.html) algorithm is an API that enables the traversal of the DOM tree of any website. It is used with JavaScript to enable the code to traverse the DOM tree in order to visit any needed

node for reading or update as a try to change the website interface dynamically. We use this API in our work in order to search the nodes to search for the most suitable node that matches the retrieved annotated text for some annotation in the highlight process. Figure 7 represents a sample DOM tree traversed by the *TreeWalker* algorithm. However, both the *TreeWalker* algorithm and the similarity between annotated texts and the websites' components will be discussed in the coming sections.

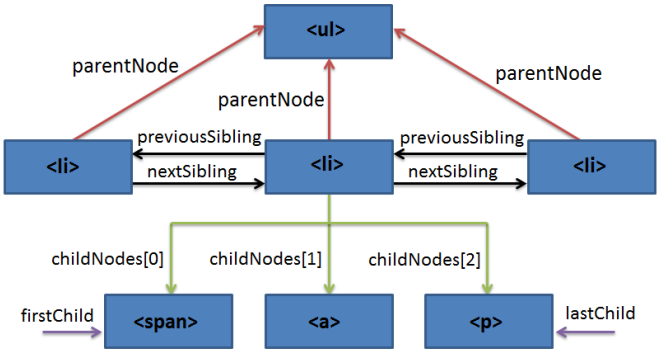

Fig. 7. Sample of DOM tree traversed by TreeWalker.

B. Node JS

This part is mainly an interface between the *Google Extension* and the *Database* parts of the system. Upon the creation of some annotation, the *Google Extension* sends all related data to the *Node JS* that contacts the database server in order to save the annotation. On the other side, this part of the system is responsible to contact the database another time to retrieve all annotations with their related data for a given logged username and the URL being visited to be all sent to the *Google Extension*. These activities are previously illustrated and depicted in the sequence diagram appears in Figure 6.

C. Database

We use *MongoDB* as a DBMS in order to manage the annotations' storing and retrieving. It is a source-available cross-platform document-oriented DBMS classified as a NoSQL database program. It supports field, range query, and regular-expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions. Queries can also be configured to return a random sample of results of a given size.

D. Python Code and TreeWalker

This component is the key part of our proposed framework in solving the annotation of dynamic contents of websites. Both *Python Code* and the *TreeWalker* collaborate in order to provide a NLP-based solution for the problem. The process starts when some user visits a given URL and asks the annotation system to retrieve all annotations previously submitted for that URL. As a result, the annotation system contacts the database and retrieves all URL annotations as discussed in the previous sections. Now, the functionality of the solution starts upon the retrieval of all annotations and their related data. The *TreeWalker* algorithm implemented in the *Google Extension* takes the first annotation and starts visiting all nodes in the DOM tree of the current URL. Upon each visit, the algorithm contacts the *Python Code* in order to compare the textual content of the current node and the highlighted text of the current annotation. Upon a value of similarity, the *Python Code* decides if both (the textual content of the current node and the highlighted text) are the same or not. If they are the same, then the *Google Extension* highlights the textual content of the current node and links the current annotation with it. If not, the *TreeWalker* continues its traversal to the next DOM node. An annotation is lost if the *Python Code* is unable to find the most similar textual content of all visited DOM nodes.

The textual similarity computation process implemented in the *Python Code* can be summarized as follows:

1) Cleans both textual contents of the current DOM node and the highlighted text (retrieved from the database) from the stop words.
2) Stems the remaining keywords in order to compare the roots of words instead of their different forms.
3) Executes the textual similarity computation between both texts ($Threshold >= 80\%$).

Figure 8 below illustrates the whole process. The execution starts with loading the first annotation from the database, the *TreeWalk* then visits the first textual node of the website's DOM tree. Both highlighted text and textual data of the node are sent to the *Python code* to compute the similarity between them. If the amount of similarity is above some threshold, then the node is added to a sorted list of nodes, if not, the *TreeWalker* algorithm continues to the next node. With the last tested node, the sorted list contains a list of similar texts to the highlighted text each of which has its own similarity amount. With a decreased sort of this list, the top node will be the most similar to the highlighted text, in which the *Google Extension* JavaScript code highlights the textual content of the node and links the current annotation. After that, the same steps are executed for the rest of the annotations of the website. Of course, the textual processing (cleaning and stemming) is executed on both the highlighted text of the current annotation and the textual content of the current node before computing the similarity measure.

V. Time Complexity Analysis

In this section, we perform a complexity analysis for the proposed algorithm. Suppose a given website S has a set of annotations of size n dented by $A = \{a_1, a_2, a_3, \ldots, a_n\}$ annotations submitted by some users and the website has a set of DOM nodes N of size m denoted as $N = \{n_1, n_2, n_3, \ldots, n_m\}$ and that the time needed to traverse the

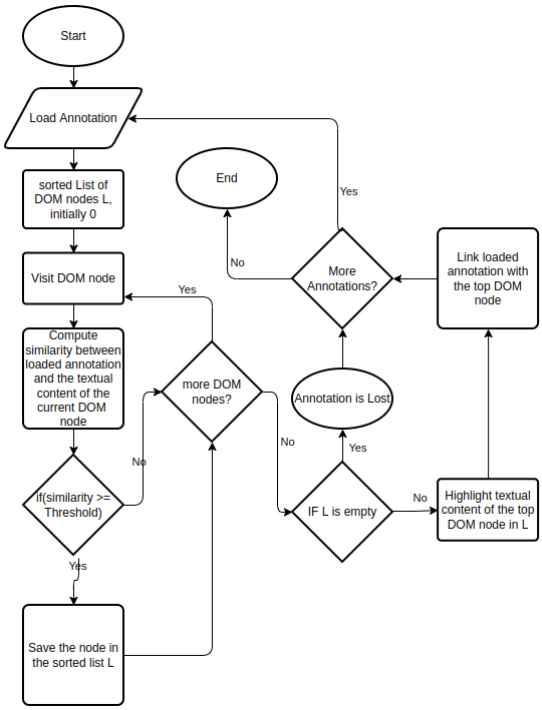

Fig. 8. Flowchart of highlighting the most suitable textual content with some annotation.

DOM tree by the *TreeWalker* algorithm is given by $T(n_i)$ where i is the i_{th} node and $0 \leq i \leq m$ with the assumption that the time needed to load all annotations for a given website is neglected since the load is executed one time per URL. In this case, all time spent goes to the evolving of annotations over all nodes and is given by the following formula:

$$T(A,N) = \sum_{i=1}^{k} \sum_{j=1}^{M} T(n_i) + sim(a_i, n_k) \tag{1}$$

where $sim(a_i, n_k)$ is the time needed to compute the similarity between the annotation a_i and the textual content of the node n_k. However, the time needed to execute the similarity is constant for all annotations and nodes which leads to substitute $sim(a_i, n_k)$ by some constant $C1$. Moreover, considering that the DOM tree for a given website is balanced, leads to the assumption that the traversal time for a given node is also constant $C2$. By this, the formula becomes:

$$T(A,N) = \sum_{i=1}^{k} \sum_{j=1}^{M} C1 + C2 \tag{2}$$

that can be reduced to:

$$T(A,N) = \sum_{i=1}^{k} \sum_{j=1}^{M} C \tag{3}$$

The summation $\sum_{j=1}^{M} C$ equals to CM that leads to the equation:

$$T(A,N) = \sum_{i=1}^{k} CM \tag{4}$$

and the same for this to have CNM which $\in O(n^2)$ as a complexity time.

VI. Experimental Tests

This section contains a description of the comprehensive test we conducted in order to measure the amount of collaboration between users with and without our proposed solution. To do this, we asked 20 users from different disciplines to involve in the test that lasts for 5 days [14]. Before beginning the test, we prepared a set of dynamically content change websites (10) and we disabled the proposed solution in the implemented annotation system then we asked the users to start submitting annotations to these websites with replies to each other. At the end of this stage, we counted the numbers of submitted and lost annotations:

1) Submitted (with replies): 532.
2) Lost (with replies): 452

After this, we enabled the proposed solution in the annotation system and again asked the users to start another annotation process between each other and on the same websites. At the end of this stage, we also counted the numbers of submitted and lost annotations:

1) Submitted (with replies): 635.
2) Lost (with replies): 76

Figure 9 depicts the degradation in the lost annotations after enabling the proposed solution.

Despite the difference between lost annotations without and with our proposed solution reflecting an improvement in the intended collaboration between users in the test, we asked the users to fill two questionnaires that contain the following questions, again before and after the proposed solution (We counted the ”yes” answers):

1) Q1: Did you lose important information? (Before:18, After:3).
2) Q2: Did you get useful replies for your annotations? (Before:3, After:16).
3) Q3: Did you have a complete discussion between you and others regards some topic? (Before:1, After:20)
4) Q4: Are you satisfied with the annotation process? (Before:0, After:19).

In order to quantify the amount of collaboration between users depending on the values of these four questions, we computed the ratio of answers per question after/before proposing the solution [15], [16]. Suppose that Y_{after} = number of yes

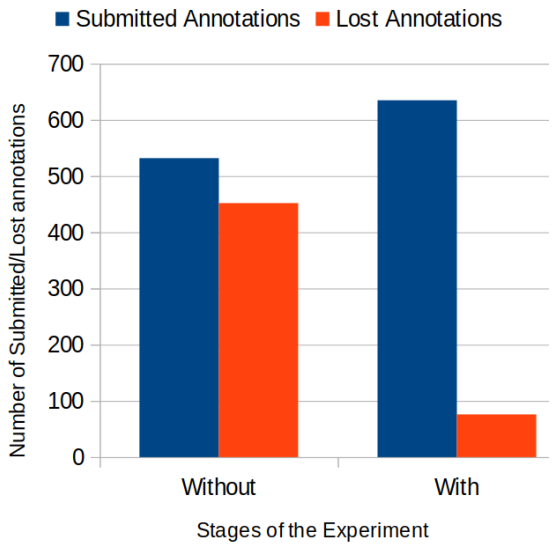

Fig. 9. Number of submitted/lost annotations without and with the proposed solution.

after proposed solution and Y_{before} = number of yes before proposed solution, then:

$$Ratio = \frac{Y_{after}}{Y_{after} + Y_{before}} \tag{5}$$

According to the formula, we got:

1) Amount of information loss: $\frac{3}{3+18} = 14\%$.
2) Useful replies: $\frac{16}{16+3} = 84\%$.
3) Complete discussion: $\frac{20}{20+1} = 95\%$
4) Satisfaction: $\frac{19}{19+0} = 100\%$

It is clear from these values that we got an improvement in collaboration between users.

VII. Conclusion & Future Works

In this paper, we have presented a comprehensive annotation framework to resolve the problem of annotating dynamic web pages. The similarity between the present content of the web page and the previously created annotations is computed following an NLP algorithm. This similarity is further exploited to link the existing annotations in the database to the proper contents of websites. Consequently, users can have discussions and hence better collaborate with each other as the experimental tests emerged.

For future works, we are planning to include the move from dynamic textual contents of websites to dynamic images. In the case of the removal of some images, the future work solution searches for the most similar image that exists on the website to link the annotations with. Of course, this needs some image processing algorithms to be involved.

References

[1] S. T. Kadam and S. Bajpai, "Development of web annotation technique for search result records using web database," in *2015 International Conference on Computing Communication Control and Automation*, pp. 348–352, 2015.

[2] T. Thindwa, W. Chawinga, and G. Dube, "Information-seeking behaviour of security studies students: A case study," *SA Journal of Information Management*, vol. 21, 05 2019.

[3] E. Asmar, S. Salahat, F. Zubdeh, and A. Hawash, "Enhancing users collaboration by vocal annotations," in *2021 18th International Multi-Conference on Systems, Signals & Devices (SSD)*, pp. 845–851, 2021.

[4] D. Biber and E. Finegan, "Styles of stance in english: Lexical and grammatical marking of evidentiality and affect," *Text-interdisciplinary journal for the study of discourse*, vol. 9, no. 1, pp. 93–124, 1989.

[5] A. Hawash, D. Avola, P. Bottoni, M. Antico, K. Kanev, and F. Parisi Presicce, "An interactive tool for sketch-based annotation," *Japanese Journal of Applied Physics*, 12 2015.

[6] D. Avola, P. Bottoni, and A. Hawash, "Supporting group collaboration in an annotation system," *J. Vis. Lang. Comput.*, vol. 41, p. 22–40, aug 2017.

[7] C. Beck, H. Booth, M. El-Assady, and M. Butt, "Representation problems in linguistic annotations: ambiguity, variation, uncertainty, error and bias," in *14th Linguistic Annotation Workshop*, pp. 60–73, 2020.

[8] S. Razon, J. Turner, T. Johnson, G. Arsal, and G. Tenenbaum, "Effects of a collaborative annotation method on students' learning and learning-related motivation and affect," *Computers in Human Behavior*, vol. 28, pp. 350–359, 03 2012.

[9] J.-H. Park and C.-W. Chung, "Semantic annotation for dynamic web environment," in *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, (New York, NY, USA), p. 353–354, Association for Computing Machinery, 2014.

[10] V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna, "Semantic annotation for knowledge management: Requirements and a survey of the state of the art," *Journal of Web Semantics*, vol. 4, no. 1, pp. 14–28, 2006.

[11] Y. Liao, M. Lezoche, H. Panetto, and N. Boudjlida, "Why, where and how to use semantic annotation for systems interoperability," 06 2011.

[12] M. Farouk, S. El-Beltagy, and M. Rafea, "On-the fly annotation of dynamic web pages.," pp. 327–332, 01 2005.

[13] I. Navas-Delgado, N. Moreno-Vergara, A. Gomez-Lora, M. del Mar Roldan-Garcia, I. Ruiz-Mostazo, and J. Aldana-Montes, "Embedding semantic annotations into dynamic web contents," in *Proceedings. 15th International Workshop on Database and Expert Systems Applications, 2004.*, pp. 231–235, 2004.

[14] J. Martínez-Mesa, D. González-Chica, J. Bastos, R. Bonamigo, and R. Duquia, "Sample size: how many participants do i need in my research?," *Anais brasileiros de dermatologia*, vol. 89, pp. 609–615, 07 2014.

[15] M. Coccia, "Metrics to measure the technology transfer absorption: analysis of the relationship between institutes and adopters in northern italy," *Int. J. Technology Transfer and Commercialisation*, vol. 4, no. 4, 2005.

[16] H. B. Santoso, M. Schrepp, R. Isal, A. Y. Utomo, and B. Priyogi, "Measuring user experience of the student-centered e-learning environment.," *Journal of Educators Online*, vol. 13, no. 1, pp. 58–79, 2016.