Automating Arabic Tags Creation for Annotating Web Contents

Jana Alhasan, Shahd Sulaiman, Batool Barham, Amjad Hawash*
Information & Computer Science
An-Najah N. University
Nablus, Palestine
jannahasan@hotmail.com, shahdhatem560@gmail.com, batoolbarham11@gmail.com, amjad@najah.edu

***Abstract*—Web Annotation becomes an important collaboration technique for users of the web. In order to increase universal collaboration, it is vital for annotators to contact the most related people that share the same interests. Although annotations themselves can be used to exchange ideas and experts, the improper writing of their attached notes could decrease the intended collaboration due to the lack of expressing ideas clearly. Adorning annotations with proper tags makes it easier for annotators to express their feelings, emotions, and interests through their submitted annotations. The ability to use the tags in searching for annotations leads to reaching the most related tags to one's interests and this indeed increases changing ideas between annotators and hence their collaboration. However, expressing the ideas in annotations with proper tags is not an easy task for most annotators. On the opposite side, attaching improper tags loses the soul and the intention of creating annotations which decreases the amount of collaboration. This work is related to automating the process of tags' suggestion by studying the texts selected by annotators over the web. Suggesting proper tags takes into account the part of speech of the annotated texts. The experimental results conducted in the work are related to finding the most suitable threshold for the percentage number of tag suggestions.**

***Index Terms*—Web Contents, Annotations, Tags Suggestion.**

I. Introduction

Authors of websites try to include different visual contents in order to make websites more attractive to users [1]. This increases the number of visitors that definitely raises up the importance of websites by increasing their page ranks. Several ways are used to boost the value of websites: (1) increase website usability by different optimization services, (2) work on website advertising, (3) includes metadata to be noticed by search engines, (4) insert right keywords in the website, (5) exploit the Content Delivery Network (CDN), (6) increase website worth through testimonials and star ratings, and (7) invites others to guest blog on site.

Web annotation is defined as adding extra information to website contents by attaching textual, vocal, and sketch annotations to websites' contents. Adding annotations leads to enriching these contents and enables conducting online collaboration between annotators. However, adding annotations to the contents of some websites has several advantages related to a better understanding of the website's materials, increasing the collaboration between annotators, and increasing the critical thinking about the contents of the annotated websites [2].

Keywords and keyphrases can be defined as sequences of one or more words extracted from documents that provide important and descriptive information about their contents in a way such that these words and phrases may serve as either an indicative summary or as a document metadata. Both can help the reader in their search for relevant information [3]. Keywords can be used to facilitate the more efficient searching of information and to help the reader to make a decision about whether it is necessary to read a document or not.

Several types of annotations are used to add extra information to website contents. Textual, vocal, and sketch annotations make it easier for web users [1] to choose the best way to express their thoughts and believes [4], [5]. The various types of annotations enrich the contents of websites and make it easier for users to collaborate by exchanging their ideas by submitting related annotations with their replies.

While creating annotations and attaching them to websites' contents is a famous method these days in expressing users' feelings and thoughts, searching for proper annotations that match one's interests still needs some improvements [6]. The main problem for annotators is *"how to find the proper annotations that match their interests?"* in order to share ideas with other people having the same concerns. Finding the proper annotations makes it easier for annotators to share their ideas and feelings with others regards some website content that improves both annotators and websites, evenly.

Several ways are implemented in annotation tools to enhance the finding of the proper annotations on the web. Linking annotations with ontologies, and introducing groups to annotation systems so that a group contains shared users of interests are two ways of increasing the probability of finding the most related annotations to someone's interests and hence conducting some collaboration between annotators [7]. However, augmenting annotations with some related tags improves the search for the most suitable annotations [8]. Adding expressing tags to annotations makes it easier for annotators to find the most suitable annotations by doing a tag-based search that retrieves the related annotations.

Several techniques are used to better express ideas either by annotations or any other means of disclosing thoughts. Planning for the idea, using imagery, answering questions, and practicing are a set of advice to better express ideas and feelings [9]. The deficiency in expressing emotions and ideas by synthesizing textual annotations minimizes the collaboration between annotators who seek those who share the same interests.

Corresponding Author:amjad@najah.edu

[1]Users and annotators will be used exchangeable in this work.

Creating vocal annotations instead of textual ones helps in better expressing ideas since the recorded voice may contain illustrative tones that help listeners to understand the soul of published annotation [4]. However, searching for annotations is not an easy task if they are vocal ones. On the other side, attaching tags to annotations is considered one of the techniques that help in clarifying the intention of some annotations and helps in searching for annotations [10].

Creating suitable tags that express the intent of the annotations is not an easy process for most novice annotators especially if the annotated material is specialized and not related to general topics. The failure in creating suitable tags leads to the failure in expressing the sole of the annotations and this will lead to a two-sided problem: from the posting side, the ability to recommend annotations to others will be decreased, and from the reading side, finding the proper annotations will be decreased or non-relevant annotations will be retrieved [11]. The idea of synthesizing the suitable tags leads to adding semantics to the contents of the web and hence adding extra useful information that leads to finding the most related annotations.

Automating the creation of tags rather than the manual technique has several advantages such as getting them faster, cheaper, and in high quality [12]. Several algorithms were invented to study textual materials to create the most suitable related tags. Artificial Intelligence techniques (NLP, machine learning, neural networks) and Information Retrieval algorithms are all examples of such algorithms.

Our contribution to this work can be summarized as follows:

1) Implementing a textual-based simple annotation system that enables annotators of the web to create textual annotations for textual contents.
2) Attaching annotations with tags created automatically by studying the annotated texts in order to extract their most suitable tags where a ranked list of tags is suggested to users depending on the values of scores and weights per tag.
3) Conducting comprehensive experimental tests in order to measure the amount of accuracy of the tags suggestion algorithm and to fix the value of a threshold that represents the amount of percentage of created tags.

After the user highlights some text in a website, the invented algorithm studies the selected text and extracts its most related tags all ranked according to their parts of speech to be added to the annotation popup window. The algorithm takes the highlighted text as input and extracts the most important terms to be used as a suggested list of tags. However, the user has the ability to make some updates to the list of tags by either adding or removing some tags. The algorithm harnesses the *tf-idf* technique to generate the list of tags with the ability for the user to specify a threshold value to control the percentage of generated tags.

The rest of this paper is organized as follows: Previous work is proposed in Section II. Section III discusses the system architecture of the implemented work while Section IV is related to the conducted experimental test. Finally, Section V concludes this paper.

II. Related Works

Attaching textual, vocal, and drawing sketches to websites' contents is recently considered one of the important techniques used to exchange ideas between web users and hence conducting some kind of collaboration between them [4], [5], [8].

Several tools have been invented to create annotations of web content. Diigo [2], A.nnotate [3], Bounce App [4], Markup.io [5] and Filestage [6] are all examples of famous annotation tools used to add sticky notes attached with texts and images of websites.

However, annotators in general, find it hard sometimes to express ideas in their minds as annotations in the process of doing some online discussions with others [13]. Expressing ideas correctly clarify their opinion regards topics under discussion and hence enhances their collaboration.

The work of [14] is related to harnessing an NLP-based technique in classifying textual data by determining the type of entities by calculating the semantic proximity of vectors obtained using neural network language models. Their approach has the advantage of the low laboriousness of text corpus preparation in comparison with traditional methods of learning with a teacher and methods based on rules.

ABNER (A Biomedical Named Entity Recognizer) is the project described in [15] and related to an open source software tool for molecular biology text mining. It is a machine learning system using conditional random fields with a variety of orthographic and contextual features. It has an intuitive graphical interface and includes two modules for tagging entities (e.g. protein and cell line) trained on standard corpora.

Auto-tagging of research articles is the work of [16] where the authors of the work invented an algorithm of two stages: classification and tag selection. The classification process involves automatic keyword extraction using the RAKE algorithm which uses a keyword–score matrix. Cosine similarity is used for classifying the articles into corresponding domains utilizing the extracted keywords. Tag selection concentrates on the selection of proper tags for the research article. The graph-based method is used for tag selection by ranking the tags within each class.

Creating tags for the Arabic language-based online content is considered not an easy task. The Arabic language has some special complex situations with respect to the semantic web. This is due to the nature of language itself that can be summarized in connected Arabic letters, the change in meaning for a word because of the usage of dialectics, and the ambiguity of some words that have several meanings like the word رفعت that could mean a person name or the verb hold up [17].

However, several works are conducted to work with Arabic contents in terms of automatic Arabic tags generation. The work published in [18] is related to a general-purpose annotation tool for Arabic text with a focus on readability annotation. *MADAD* tool helps in overcoming the problem of

[2]https://www.diigo.com/
[3]http://a.nnotate.com/
[4]https://www.bounceapp.com/
[5]https://www.markup.io/
[6]https://filestage.io/

lack of Arabic readability training data by providing an online environment to collect readability assessments on various kinds of corpora. Moreover, the tool supports a broad range of annotation tasks for various linguistic and semantic phenomena by allowing users to create their customized annotation schemes. The tool is a web-based tool, accessible through any web browser; the main features that distinguish *MADAD* are its flexibility, portability, customizability, and its bilingual interface (Arabic/English).

The work of [19] used *word2vec* model to build Arabic vector representations of words that brings extra semantic features to help build clusters of semantically related words. This involves text pre-processing, creating word vectors using *word2vec* model, generating the classification model, and creating word clusters using the Pipeline method and Extra tree classifier. The term frequency matrix and learn to classify the vectors with Extra Tree classifier, then classifying and predicting the training data into the pre-defined categories. The authors of the work have implemented the model and performed a set of experiments with results showing the effectiveness of the model to create word clusters from a large plain Arabic text. The classification results show that the extracted features from the word vectors have empowered the classification models and achieved accuracy, precision, recall and F-measure higher than 85%.

Our contribution in this work is related to implementing a *tf-idf* technique with the help of scoring the extracted terms from the highlighted contents of the web in order to suggest their most relative tags and attach them to annotations submitted by a simple textual based annotation system developed to clarify the idea of tags automation to annotate Arabic online contents with the ability to specify the amount of percentage of created tags that can be fixed by annotators themselves.

III. System Architecture

In this section, we describe our client-server architecture system of creating annotations and adorn them with the proper tags. Figure 1 depicts the main system components in which the client side is mainly composed of Google Extension that contains special JavaScript code to be used for highlighting textual contents of the web and to create annotations to be submitted to the server side of the system to be saved and retrieved upon requests. However, the server side is composed of PHP code, Python scripts, and a dedicated database to save the submitted annotations with their related data. In the following subsections, we describe each part in detail.

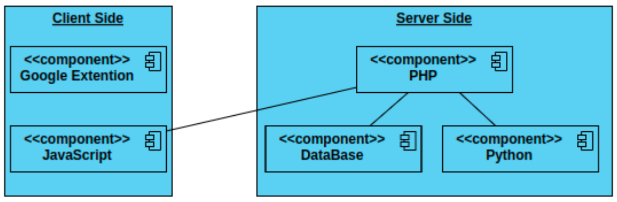
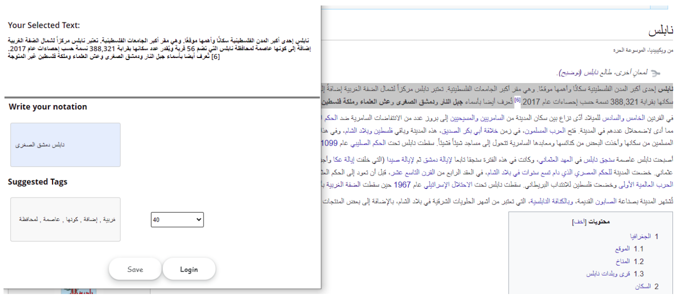


Fig. 1. System Components.

A. Client Side

This part of the system is considered the interface that can be used by annotators to interact with the system. It is composed of several parts described in the following:

1) Google Extension: that contains a special JavaScript code to handle the interactions with annotators. Upon the highlight of some web textual content, a pop-up window appears containing the selected text with two spaces one for writing the annotation itself and the other to be used to write the tags or to contain the automated tags created by the server side of the tool. When the tool is asked to generate a list of tags, the JavaScript code interacts with the server side to generate a list of tags for the selected text. This process will be described in detail in the coming sections. Figure 2 depicts the process of creating an annotation where the selected text is copied inside the pop-up window and sent to the server side to extract the suggested tags. The 40 value represents the percentage value of retrieved tags.


Your Selected Text:

نابلس إحدى أكبر المدن الفلسطينية سكانًا وأهمها موقعًا. وهي مقر أكبر الجامعات الفلسطينية. تعتبر نابلس مركزاً لشمال الضفة الغربية إضافة إلى كونها عاصمة لمحافظة نابلس التي تضم 56 قرية ويُقدر عدد سكانها بقرابة 388,321 نسمة حسب إحصاءات عام 2017. تُعرف أيضا بأسماء جبل النار ودمشق الصغرى وعش العلماء وملكة فلسطين غير المتوجة [6]

Write your notation

نابلس دمشق الصغرى

Suggested Tags

غربية , إضافة , كونها , عاصمة , لمحافظة

40

Save Login

Fig. 2. Annotation creation with suggested tags.

Figure 3 depicts the process of creating a new annotation with a list of tags generated by the server part of the system. The scenario starts by executing the method *newAnnotation(selectedText)* defined in the object *Google Extension*. The later executes the method *send(slectedText)* defined in the object *PHP Engine)* then the method *gatTags(selectedText)* on the object *Python Engine)* that executes a self defined method *extractTags(selectedText)* in order to study the selected text to generate a list of related tags. This list called *tags[]* is then returned to the object *PHP Engine* to be returned to the object *Google Extension* that in terns displays the list to the user. The user is then notified by the method *notifyAnnotation(tags[])* and then examines the generated list of tags by updating the list manually if s/he wishes. When the user agrees with the tags, a series of *saveAnnotation()* method is executed starting from the user up to *PHP Engine)* passing the necessary annotation data as a parameter. The *Database Server* object is then saves the annotation alongside with its related data in a dedicated database and then notifies both *PHP Engine* and *Google Extension* with the return value *annotationSaved=true* and finally the used is notified by the method *notifyUser()* that the annotation is saved completely.

2) Website: that enables users to create and update their accounts as well as interact with their own annotations. The website supports users with a set of services such as:

1) Listing all annotations submitted by some user.
2) When an annotation is clicked, the user is directed to the corresponding website that contains that annotation.

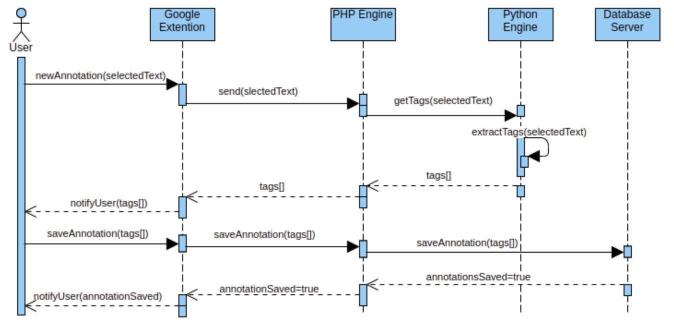
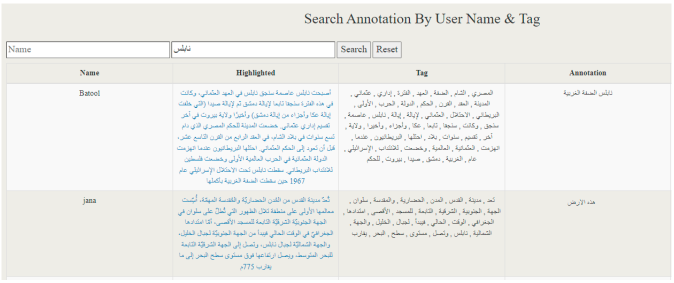


Fig. 3. Sequence diagram for submitting an annotation with its related tags.

3) A form to search for annotations, tags, and users.

The searching facility enhances users' collaboration because it can be used to search for some tags and the system retrieves all annotations related to searched tags. Upon the clicking of some annotation, the system opens the related website in a new window and highlights all texts previously selected by all users for that website. When the user clicks one of the highlighted texts, the system retrieves all of its related annotations. We implemented the ability to display others' annotations in order to increase the collaboration between users by reading others' notes and submitting replies to their notes attached to the same annotation to be all saved in the database to be viewed by other annotators. This creates some kind of collaboration between users in terms of submitting consecutive replies for the submitted annotations. By this, the users are able to discuss some idea that appears in the highlighted texts by replying to each other with their thoughts and ideas. Of course, each reply is attached with its corresponding list of tags. Figure 4 displays a set of annotations as a result of searching for some tag.


Search Annotation By User Name & Tag

Name	نابلس	Search	Reset

Name	Highlighted	Tag	Annotation
Batool	أصبحت نابلس عاصمة سنجق نابلس في العهد العثّماني، وكانت في هذه الفترة سنجقا تابعا لإيالة دمشق ثم لإيالة صيدا (التي خلفت إيالة عكا وأجزاء من إيالة دمشق) وأخيرًا ولاية بيروت في آخر تقسيم إداري عثّماني. خضعت المدينة للحكم المصري الذي دام تسع سنوات في بلاد الشام، في العقد الرابع من القرن التاسع عشر، قبل أن تعود إلى الحكم العثّماني. احتلها البريطانيون عندما انهزمت الدولة العثّمانية في الحرب العالمية الأولى وخضعت فلسطين للانتداب البريطاني. سقطت نابلس تحت الاحتلال الإسرائيلي عام 1967 حين سقطت الضفة الغربية بأكملها	المصري , الشام , الضفة , العهد , الفترة , إداري , عثّماني , المدينة , العقد , القرن , الحكم , الدولة , الحرب , الأولى , البريطاني , الاحتلال , العثّماني , لإيالة , إيالة , نابلس , عاصمة , سنجق , وكانت , سنجقا , تابعا , عكا , وأجزاء , وأخيرا , ولاية , آخر , تقسيم , سنوات , بلاد , احتلها , البريطانيون , عندما , انهزمت , العثّمانية , العالمية , وخضعت , للانتداب , الإسرائيلي , عام , الغربية , دمشق , صيدا , بيروت , للحكم	نابلس الضفة الغربية
jana	تُعدّ مدينة القدس من المُدن الحضاريّة والمُقدسة المهمّة، أُسّست معالمها الأولى على منطقة تلال الظهور التي تُطلّ على سلوان في الجهة الجنوبيّة الشرقيّة التابعة للمسجد الأقصى، أمّا امتدادها الجغرافيّ في الوقت الحالي فيبدأ من الجهة الجنوبيّة لجبال الخليل، والجهة الشماليّة لجبال نابلس، وتصل إلى الجهة الشرقيّة التابعة للبحر المتوسط ويصل ارتفاعها فوق مستوى سطح البحر إلى ما يقارب 775م	تعد , مدينة , القدس , المدن , الحضارية , والمقدسة , سلوان , الجهة , الجنوبية , الشرقية , التابعة , للمسجد , الأقصى , امتدادها , الجغرافي , الوقت , الحالي , فيبدأ , لجبال , الخليل , والجهة , الشمالية , نابلس , وتصل , مستوى , سطح , البحر , يقارب	هذه الارض

Fig. 4. Listing annotations with a shared tag.

B. Server Side

The major contributions of this work go in this section. The server side is implemented by *PHP* and *Python* scripts linked to the MySQL database server. The PHP code is used as an intermediary between the Google Extension and the Python code. All of the database transactions that are related to saving and retrieving annotations with their related data are executed by PHP code. However, the Python code contains the algorithms that represent our main contributions of automating the creation of tags to be attached with their corresponding annotations. The following subsections describe the major parts of the server side.

1) Database: This part of the system is used to save annotations with their corresponding data managed by MySQL using a dedicated server. Figure 5 represents the Entity-Relationship diagram for the entities used to build up the relational database. The entity *User* is used to save data about users and has a one-to-many relation with the *Annotation* entity using the primary key *userID*. The *Annotation* entity is used to save data for the submitted annotations and has a many-to-many relationship with the *Tag* entity that is used to save the list of tags per annotation. The *Annotation* entity also has a reflexive relationship to itself in order to save the replies annotation, this represented by the existence of *Replies* entity.

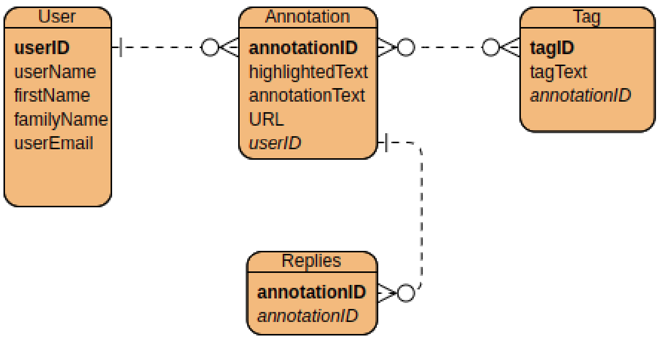


Fig. 5. Entity Relationship Diagram.

2) Python Scripts: This part of the server side, and as mentioned before, is responsible for most actions taking place on the server side. We will discuss the services of the code in the following:

a) Assigning scores to Parts of Speech: The main purpose here is to assign a score value for each part of speech approved in this work *(Noun, Verb, Adjective, Adverb)*. These scores will be used for further computation in order to better enhance the process of tags suggestion. During this part (executed one time in order to fix the values of scores), we randomly used 20 different Arabic documents, each of which is composed of a set of paragraphs. Initially, each document is processed by:

i) applying a python library called *pyarabic* [7] that provides basic functions to manipulate Arabic letters and text, like detecting Arabic letters, Arabic letters groups and characteristics, remove diacritics, etc.

ii) Removing stop words where a list of Arabic stop words are taken from *nltk* [8] library (natural language toolkit).

to remove its stop words to have a list of terms to be fed to a Java library to determine the terms of each part of speech. We used the Java library *Stanford Log-linear Part-Of-Speech Tagger* [9] that supports the Arabic language. Moreover, the same documents

[7]https://pypi.org/project/PyArabic/
[8]https://www.nltk.org/
[9]https://nlp.stanford.edu/software/tagger.shtml

are examined manually by 5 Arabic language part of speech experts who were asked to parse the documents and extract a list of the most important words with their corresponding parts of speech. This process is executed gradually, that is we started with the first document that is parsed by the 5 experts and fed to the Java tool. We asked the experts to assign a score for each one of the parts of speech mentioned *(Noun, Verb, Adjective, Adverb)*. At the end of processing the first document, we got 24 lists of terms described as follows:

i) Each expert generated 4 lists of terms, one for each part of the speech. In total, we had 20 lists for the 5 experts.
ii) 4 lists of terms, one for each part of speech all generated by the Java tool.

After generating the previous lists, we computed the amount of Accuracy per each part of speech between the 4 lists generated by the 5 experts and the one generated by the Java tool according to Equation 1:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

where

i) *TP (True Positive)*: is the number of correct terms between the two lists.
ii) *TN (True Negative)*: is the number of non-terms excluded in the two lists.
iii) *FP (False Positive)*: is the number of terms predicted by the tool and excluded by experts.
iv) *FN (False Negative)*: is the number of terms excluded by the tool and included by the experts.

As a result, we got 4 different values of accuracy per expert. In order to have one value of the accuracy per each part of speech, we computed the average value for all accuracy values per each expert to get 4 different values of Accuracy.
Suppose the following:

i) E_i: is the i_{th} expert.
ii) $E^k_{i,j}$: is the j_{th} term for the i_{th} expert for the k_{th} part of speech where $k \in \{noun, verb, adverb, adjective\}$.
iii) T^k_s: is the s_{th} term for the Java tool for k_{th} part of speech where again $k \in \{noun, verb, adverb, adjective\}$..

and suppose that $A^k(E_{i,j}, T_s)$ is the accuracy value computation operation (denoted by $\blacklozenge$) between the set of expert terms $E_{i,j}$ and the set of tool terms T_s for the part of speech k according to equation 1. So, $A^k(E_{i,j}, T_s) = E_{i,j} \blacklozenge T_s$ where again $1 \geqslant i \geqslant 5$, j goes between 1 and the total number of terms found by the expert i for the k_{th} part of speech that is in $\{$noun, verb, adverb, adjective$\}$ and s goes between 1 and the number of terms found by the tool.
According to this, we will have 4 accuracy values per expert. To compute the average values of accuracy per each part of speech we use the following Equation:

$$Av(k) = \frac{\sum_{i=1}^{5} A^k(E_i, T_s)}{5} \quad (2)$$

where $Av(k)$ is the average value for all accuracy values of all experts for the part of speech k. Figure 6 depicts the extraction of parts of speech by both the experts and the Java script for a sample document as well as the average and accuracy computation all numbered chronically from 1 to 4.

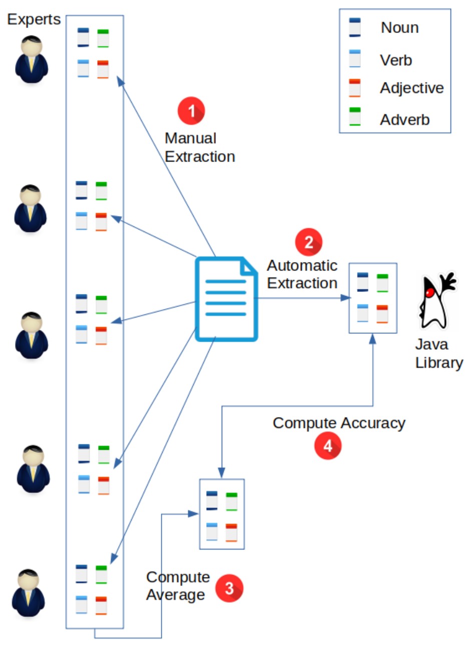


Fig. 6. Accuracy computation.

Table I contains the values of the accuracy of the 4 parts of speech between the 5 experts and the Java tool for a sample document.

Expert #	**1**	**2**	**3**	**4**	**5**
Name	0.95	0.93	0.96	0.94	0.9
Verb	0.85	0.86	0.84	0.85	0.85
Adverb	0.62	0.64	0.63	0.68	0.7
Adjective	0.78	0.85	0.81	0.76	0.75

TABLE I
VALUES OF THE ACCURACY OF THE 4 PARTS OF SPEECH BETWEEN THE 5 EXPERTS AND THE JAVA TOOL FOR A SAMPLE DOCUMENT.

At this point, we computed the amount of accuracy for each part of speech for all 5 experts against the tool for the first document. We asked the experts during this iteration to manually assign some scores for each part of speech according to the importance of parts of speech they found in the first document. Suppose S_k denotes the score value of the t_{th} part of speech. We need to combine the number of accuracy values and the number of scores for all the parts of speech in order to get a new list of scores to be used (or updated) by the experts for the next document. Recursively, this will be repeated for all of the documents. That is, the scores of the i_{th} document will be used to compute the scores of the

i_{th+1} document. So, the computation of scores is done using the following equation:

$$S_i^k = \frac{Av_i^k + S_{i-1}^k}{2} \tag{3}$$

where S_i^k is the current score value of the k_{th} part of speech, Av_i^k is the average accuracy value for the current document i for the k_{th} part of speech, and S_{i-1}^k is the score of the k_{th} part of speech of the previous document. At the end of examining each document, we took the average value between the two values of accuracy and score for the current document in order to emphasize both evenly and to be sure that score was always between 0 and 1. According to this, we used to update the scores given for each part of speech after finishing the examination of each document. After examining the whole document, we got a list of parts of the speech with their corresponding scores.
After examining each document by the experts and the tool and after getting the values of accuracy, we used to update the scores of each part of speech according to the values of accuracy gained. Table II reflects the values of scores computed between the lists generated by the 5 experts and the Java tool for each part of speech for all 20 documents where the values of the last row are the final ones used in the next step.

Document #	**Noun**	**Verb**	**Adverb**	**Adjective**
1	0.95	0.93	0.92	0.93
2	0.98	0.9	0.9	0.9
3	0.98	0.8	0.8	0.91
4	0.98	0.85	0.7	0.92
5	0.97	0.86	0.71	0.89
6	0.96	0.86	0.71	0.89
7	0.96	0.84	0.76	0.89
8	0.95	0.85	0.75	0.89
9	0.95	0.83	0.71	0.88
10	0.95	0.82	0.68	0.87
11	0.95	0.79	0.68	0.89
12	0.96	0.79	0.65	0.86
13	0.95	0.79	0.65	0.85
14	0.94	0.79	0.66	0.86
15	0.93	0.75	0.66	0.85
16	0.94	0.75	0.64	0.85
17	0.95	0.73	0.65	0.84
18	0.94	0.73	0.63	0.82
19	0.94	0.72	0.61	0.82
20	0.94	0.72	0.61	0.81

TABLE II
SCORES VALUES.

Figure 7 reflects the data appear in Table II with the highest values go to nouns and adjectives at the end of iterations.
By this, we ranked the part of speech with their corresponding scores as *Noun(9.4), Adjective(8.1), Verb(7.2), and Adverb(6.1)*.

b) Terms weights: this part of the server-side is executed during the annotation submission process when the users ask the system to suggest tags for the highlighted texts. The process goes in a set of steps:

i) The textual pre-processing steps are executed by detecting Arabic letters, Arabic letter groups, and characteristics, removing diacritics, and removing stop words (as we did for the 20 documents).
ii) For the list of remaining terms, the algorithm computed their Term Frequencies (TF) and Inverse Document Frequencies (IDF).
iii) For each term, the *TF-IDF* is computed in order to assign each term with a value that represents its weight in the highlighted text.

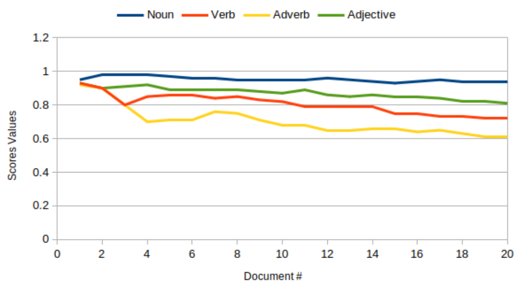


Fig. 7. Scores values for the 20 documents.

Calculating IDF for each term is computed by the following equation where N is the number of all sentences in the highlighted text and n is the number of sentences where the term t appears:

$$IDF_t = log(\frac{N}{n}) \tag{4}$$

Each term is then assigned a weight computed by the following equation where W_t is the term weight:

$$W_t = TF_t \times IDF_t \tag{5}$$

3) Suggesting Tags: the corresponding Python code of this process is executed during the annotation submission and when some user selects some text in a website to link it with some annotation. The user may ask the system to suggest some tags for the highlighted text. In this case, the code executes the textual pre-processing steps and computes the $TF-IDF$ for each term in the highlighted text as well as determining the part of speech for each term. Using the fixed values of scores for each part of speech, each term of the highlighted text is assigned a ranking value computed by multiplying the value of its score with the value of its weight in order to quantify the possibility of its tag consideration. as in equation 6

$$R_t^k = S^k \times W_t^k \tag{6}$$

where R_t^k is the rank of the term t identified as part of speech k, S^k is the score of the part of speech k and W_t^k is the weight of the term t for the same part of speech. As a result, a list of terms is generated in which each one is accompanied by a value to represent its importance in the text that can be used to rank the terms. A value of threshold appears for the users in the annotation pop-up window and is used to determine the percentage of tags users need to retrieve. According to this threshold, a list of tags with a rank greater than or equal to the threshold will be suggested to the users.

IV. Experimental Tests

A comprehensive test has been conducted in order to measure the amount of accuracy of the tags suggestion process. A set of 25 people were asked to involve the test that lasts for two weeks. The users were asked to visit websites of their interests and use the tool to create annotations. They created a set of 272 annotations (with replies) and they used the tool to suggest tags for their selected texts. During the test, we asked them to study the tags generated by the tool and to write down the number of tags either deleted from the list of those that are added to the list in order to see the number of differences between the suggested tags and the tags they need. We asked them to fix the amount of percentage of ranked tags to be 100% in order to get all of the tool suggested tags. Table III reflects the number of suggested tags by the tool, removed and added by the users, and the shared tags between the ones suggested by the tool and those approved by the users. The last column of the table represents the amounts of accuracy computed by taking the percentage between the shared and the suggested tags. By computing the average of all accuracy values we have the value 87%.

Figure 8 shows the accuracy values computed in the experiment where the range of values is above 0.8.

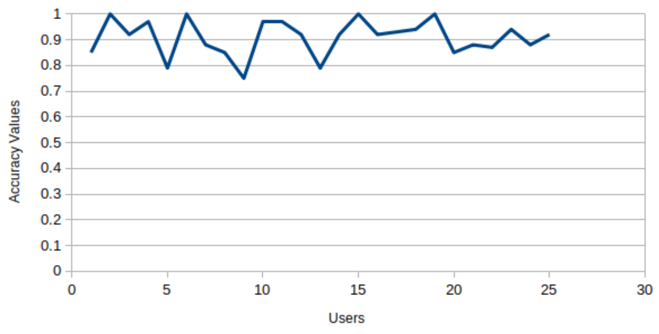


Fig. 8. Accuracy values

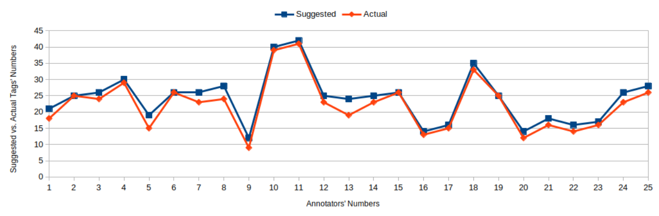


Fig. 9. Sequence diagram for submitting an annotation with its related tags.

Figure 9 represents the difference between the suggested tags by the tool and the actual ones needed by the users. We notice from the table and the figure that the two graphs are identical in high percentage which reflects a good amount of accuracy for the tool in suggesting the best tags which are also computed from the table.

	Tags				
User #	**Suggested**	**Removed**	**Added**	**Shared**	**Accuracy**
1	21	1	2	18	0.85
2	25	0	0	25	1
3	26	1	1	24	0.92
4	30	1	0	29	0.97
5	19	2	2	15	0.79
6	20	0	0	20	1
7	26	2	1	23	0.88
8	28	2	2	24	0.85
9	12	1	2	9	0.75
10	40	1	0	39	0.97
11	42	0	1	41	0.97
12	25	1	1	23	0.92
13	24	2	3	19	0.79
14	25	1	1	23	0.92
15	26	0	0	26	1
16	14	1	0	13	0.92
17	16	0	1	15	0.93
18	35	1	1	33	0.94
19	25	0	0	25	1
20	14	0	2	12	0.85
21	18	1	1	16	0.88
22	16	2	0	14	0.87
23	17	0	1	16	0.94
24	26	1	2	23	0.88
25	28	1	1	26	0.92

TABLE III
The number of suggested, removed, added, and shared tags and the amounts of accuracy.

V. Conclusion

Automating the process of tags suggestion for documents is getting more important these days. This process is important to get an abstract view of documents in which readers get an idea about the contents of documents without the need to review or read them all which minimizes efforts and times in the process of searching for information. This work is related to automating the suggestion of tags for the textual contents of the web during the process of annotations' creation and submission. The main contribution of the work is related to scoring the major parts of Arabic speech (Noun, Verb, Adjective, Adverb) in order to emphasize the most important parts of speech that better summarize texts. We developed a simple textual-based annotation system in which users highlight textual contents of the websites they prefer, and the tool computes the amounts of *TF-IDF* for each term and multiplies these values with the scores of parts of speech to get a ranked list of the most important terms to appear in the highlighted texts. The comprehensive experimental tests conducted in this work reflect a promising amount of accuracy in the process of tag suggestion.

For future works, we plan to include the annotation of video contents and implement a code to understand the talked words in order to generate the appropriate tags for video contents.

References

[1] C. Asakawa and H. Takagi, "Annotation-based transcoding for nonvisual web access," in *Proceedings of the Fourth International ACM Conference on Assistive Technologies*, Assets '00, (New York, NY, USA), p. 172–179, Association for Computing Machinery, 2000.

[2] P.-L. Patrick Rau, S.-H. Chen, and Y.-T. Chin, "Developing web annotation tools for learners and instructors," *Interacting with Computers*, vol. 16, no. 2, pp. 163–181, 2004.

[3] M. M. Al Hadidi, M. Alzghool, and H. Muaidi, "Keyword extraction from arabic text using the page rank algorithm,"

[4] 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.

[5] M. Antico, D. Avola, P. Bottoni, A. Hawash, K. Kanev, and F. P. Presicce, "An interactive tool for sketch-based annotation," *JJAP Conference Proceedings*, vol. 011604, p. 4, 2016.

[6] C. Weng and J. H. Gennari, "Asynchronous collaborative writing through annotations," in *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, (New York, NY, USA), p. 578–581, Association for Computing Machinery, 2004.

[7] D. Avola, P. Bottoni, and A. Hawash, "Users-groups matching in an annotation system: Ontological and url relevance measures," in *2014 6th International Conference on Computer Science and Information Technology (CSIT)*, pp. 100–109, 2014.

[8] A. W. Hawash, "Introducing groups to an annotation system."

[9] L. Susiana, "Improving students' ability in expressing ideas in descriptive writing by using picture series technique (a quasi – experimental research at smpn 14 bengkulu city)," *JOALL (Journal of Applied Linguistics and Literature)*, vol. 1, pp. 97–102, 02 2018.

[10] J. Zhu, C. Wang, X. He, J. Bu, C. Chen, S. Shang, M. Qu, and G. Lu, "Tag-oriented document summarization," pp. 1195–1196, 01 2009.

[11] M. G. Ames and M. Naaman, "Why we tag: Motivations for annotation in mobile and online media," pp. 971–980, 04 2007.

[12] J. Jovanovic, E. Bagheri, J. Cuzzola, D. Gasevic, Z. Jeremic, and R. Bashash, "Automated semantic tagging of textual content," *IT Professional*, vol. 16, no. 6, pp. 38–46, 2014.

[13] J. Wiebe, T. Wilson, and C. Cardie, "Annotating expressions of opinions and emotions in language," *Language Resources and Evaluation*, vol. 39, pp. 165–210, 2005.

[14] P. Mukalov, O. Zelinskyi, R. Levkovych, P. Tarnavskyi, A. Pylyp, and N. Shakhovska, "Development of system for auto-tagging articles, based on neural network," in *COLINS*, 2019.

[15] B. Settles, "ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text," *Bioinformatics*, vol. 21, pp. 3191–3192, 04 2005.

[16] M. G. Thushara, M. S. Krishnapriya, and S. S. Nair, "A model for auto-tagging of research papers based on keyphrase extraction methods," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1695–1700, 2017.

[17] K. Shaalan, S. Siddiqui, M. Alkhatib, and A. Monem, *Challenges in Arabic Natural Language Processing*, pp. 59–83. 11 2018.

[18] N. Al-Twairesh, A. Al-Dayel, H. Al-Khalifa, M. Al-Yahya, S. Alageel, N. Abanmy, and N. Al-Shenaifi, "MADAD: A readability annotation tool for Arabic text," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, (Portorož, Slovenia), pp. 4093–4097, European Language Resources Association (ELRA), May 2016.

[19] T. I. Abufayad, *Semantic Word Clustering from Large Arabic Text*. PhD thesis, The Islamic University of Gaza, 2018.