Extend an Algorithm of Auto-converting Kurdish Written Scripts in Websites: from Latin Characters to Kurdish/Arabic Characters and Vice Versa

Sardar O. Salih

College of Science, University of Duhok, Duhok, Kurdistan Region- Iraq

ABSTRACT

Nowadays, most Kurdish websites, especially news websites are required both written scripts (Latin and Kurdish/Arabic) for their users, for example, Rudaw, WAAR, NRT, etc. Therefore, designing two websites for each written script require effort, time and cost. There are web-based applications available for users not web developers to converting script manually, regardless of accuracy (misspellings), in this case, users have to input script to input box like pelk , KAL (Kurdish Academic of language) website, etc.(Jemal Nebez, 2015)(pellk, 2010). This research develops an algorithm which can auto-converting one website from its written script to its opposite for both users and web developer. for instance, if website is designed using Latin characters, this algorithm converts to its opposite programmatically. The algorithm uses JavaScript language and configured on jQuery plugin for web developers as API (Application User Interface) to use in website. In addition, there is a Firefox browser add-ons (Extension) for users (non- developers) to convert script to its opposite. This research addresses cases (irregular cases) which cause to increase misspellings, then, find solutions for each of these cases to minimal incorrect spelling. To Increase accuracy of algorithm, one website chooses as case study for analyzing algorithm output. The algorithm is tested using test methods to check errors, debugging, and accuracy.

KEYWORDS : Script, Plugin, add-ons, jQuery, API and Latin.

1. INTRODUCTION

Both Kurdish written scripts (Latin and Kurdish/Arabic) are being used depended on the territory which Kurd live there. For instance, north Kurdistan uses Latin character - Kurmanji dialect and center of Kurdistan uses Kurdish/Arabic characters - Sorani dialect (Jemal Nebez, 2015). In this research, algorithm will be designed to convert one script to its opposite programmatically, to help for who not know one of those written scripts, and help web developers to use for electronic converting, for example, developer can create one version of website and convert it to opposite. API is written in JavaScript language and used as source code to create two applications. First , is for users which help them to convert scripts named (KURDI_LATIN) Firefox add-ons (Mozil, 2010), second is a jQuery plugin named (ConvertKu-WS) which helps web developers to use in website. The reset of article is organized with sections. The first section, literature review is about current tools and algorithms available, compared with our algorithm. Then, the research methods section includes subsections, start in, converting Kurdish Arabic characters to Latin characters, with choosing irregular cases in converting and solutions for them. The second subsection is opposite, converting Latin characters to Kurdish/Arabic. After cases take considerable, then algorithm is designed with two applications Firefox addon and jQuery plugin. The third section, WAAR media website is taken for a case study and analyzing its results with testing by using applications for accuracy. The section four, is about further aspect of using algorithm. Finally, research is concluded and summarized.

Academic Journal of Nawroz University
(AJNU) Volume 7, No 3 (2018).
Received 16 Feb 2018.
Regular research paper : Published 20 June 2018
Corresponding author's e-mail : renas_rekany@yahoo.com

2. LITERATURE REVIEW

As mentioned in introduction, there are applications which are being used to convert written scripts, for instance, KAL (Kurdish Academy Language) websites are offered web-based applications to convert written script by write down text in textbox (Jemal Nebez, 2015). These web-based applications are enough to do converting regardless of algorithm accuracy. There is also a python based algorithm converts script from Kurdish/Arabic to Latin but not opposite and not doing website converting, this algorithm takes three irregular cases and its solution (Hossein Hassani & Dzejla Medjedovic, 2016). Adding more irregular cases to algorithm led to less misspellings. The output of python algorithm is taken and use for comparation with our algorithm (JavaScript algorithm) with same input to check accuracy (less misspellings). The following Figures show Kurdish/Arabic script with its converted to Latin in both algorithm.

JavaScript based algorithm, with its output Latin script

Origin Script:

ل زانكۆيا دهۆك كۆمبونەكا ئێكەتيا زانكۆييێن جيهانى هاتە ئەنجامدان دوهى 2014/12/20 ئێكەتيا زانكۆييێن جيهانى كۆمبوونەك ل دۆر وەرگرتنا قوتابيان و ئاريشێن قوتابييێن ئاوارە ودانا پلانەكێ بۆ زانكۆيێن ئەندام ل ڤێ ئێكەتيێ ل زانكۆيا دهوك ئەنجام دا..

Converted Script:

li zanikoya dhok kombuneka êketiya zanikoyiyên cîhanî hate encamdan duhî 20/12/2014 êketiya zanikoyiyên cîhanî kombûnek li dor wergirtina qutabiyan û arîşên qutabîyên aware û dana planekê bo zanikoyên endam li vê êketiyê li zanikoya dhuk encam da.

Python based algorithm, with its output Latin script

Origin Script:

ل زانكۆيا دهۆك كۆمبونەكا ئێكەتيا زانكۆييێن جيهانى هاتە ئەنجامدان دوهى 2014/12/20 ئێكەتيا زانكۆييێن جيهانى كۆمبوونەك ل دۆر وەرگرتنا قوتابيان و ئاريشێن قوتابييێن ئاوارە ودانا پلانەكێ بۆ زانكۆيێن ئەندام ل ڤێ ئێكەتيێ ل زانكۆيا دهوك ئەنجام دا.

Converted Script:

l zankoya dhok kombûneka êketya zanikoîên cîhany hate encamdan dwhy 20/12/2014 êketya zanikoîên cîhany kombûnek l dor wergrtna qutabian u arîşên qutabîên aware u dana planeky bû zanikoîên endam l vê êketîê li zankoya dhuk encam da.

By looking at **Error! Reference source not found.** Figures, there are highlight with underline words which indicate unequal output words between both algorithm. The Table below shows which words of unequal are correct in spelling in both algorithms.

Table (1) : Both algorithms' output words with corrected words

JavaScript base algorithm output words (our algorithm)	Python base algorithm output words	Correct word
Li	L	li
kombuneka	Kombûneka	kombûneka
êketiya	Êketya	êketiya
Cîhanî	Cîhany	Cîhanî
Duhî	Dwhy	Duhî
Zanikoyiyên	Zanikoîên	Zanikoyiyên
Wergirtina	Wergrtna	Wergirtina
qutabiyan	Qutabîên	qutabiyan
Û	U	û
Planekê	Planeky	Planekê
êketiyê	Êketîê	êketiyê

Look at Table above, all our algorithm (JavaScript) output words are correct in spelling comparing with Python based algorithm are not correct. There is also case study which explain in more detail the accuracy of algorithm this will be explain in section 5. In addition, there is a test to checking accuracy of algorithm, after that, two applications are made from it. First, Firefox add-on which helps users to integrate in their Firefox browser to convert any written script instantly without needs to write down any written script in text box as KAL web based application does. Second, jQuery plugin for web developers to use in their application.

3.METHODOLOGY

To convert script Latin to Kurdish- Arabic or opposite. Exceptions exist (cases). For instance, characters exist in Latin and are not in Kurdish/Arabic or opposite, see Table (**2**. Case exist with two difference Unicode code point with same character such as (ك ,ک) needs to be converted into to (k) in Latin (Unicode, 2014). Also case with double character need to be converted into one such as (ئا) into (a) in Latin (Hossein Hassani & Dzejla Medjedovic, 2016). These cases will be explained in the following subsections. The algorithm is design based on Kurdish Academic of Language alphabets table as show below.

Table (2) : Existing Kurdish alphabets by (Kurdish Academy of Language)

#	North Kurdish (Latin Kurmanjî)	Central Kurdish (Soraní - modified Arabic)
	A a	ئا ـا ا
	B b	ب ـب ـبـ بـ
	Ç ç	چ ـچ ـچـ چـ
	D d	د ـد
	E e	ه ـه ئه
	Ê ê	ئێـ ێ ـێ ـێـ ێـ
	F f	ف ـف ـفـ فـ
	G g	گ ـگ ـگـ گـ
	H h	هـ ـهـ
	I i	نا نوسری (N/A)
	Î î	ی ئی ـیـ یـ
	C c	ج ـج ـجـ جـ
	J j	ژ ـژ
	K k	ک ـک ـکـ کـ
	L l	ل ـل ـلـ لـ
	Nîne	ڵ ـڵ ـڵـ ڵـ
	M m	م ـم ـمـ مـ
	N n	ن ـن ـنـ نـ
	O o	ۆ ـۆ ئۆ
	P p	پ ـپ ـپـ پـ
	Q q	ق ـق ـقـ قـ
	R r	ر ـر
	Nîne	ڕ ـڕ
	S s	س ـس ـسـ سـ
	Ş ş	ش ـش ـشـ شـ
	T t	ت ـت ـتـ تـ
	U u	و ـو ئو
	Û û	وو ـوو
	V v	ڤ ـڤ ـڤـ ڤـ
	W w	و ـو
	X x	خ ـخ ـخـ خـ
	Y y	ی ی ـیـ یـ
	Z z	ز ـز

2.1 Converting Kurdish/Arabic Characters to Latin Characters

In this conversion, each character in Kurdish – Arabic script is converted to Latin with it's against character, for instance (ب -> b) as show in table below.

Table (3) : Kurdish/ Arabic characters against Latin charecters

KU-AR	ا	ب	د	س	ف	……
Latin	A	b	d	S	f	…….

For above characters are not so difficult to convert them due to each of them has its own against character, this can be done with following JavaScript statements:
1. var word="";
2. word = word.replace (/ب/g, "b");
3. word = word.replace (/د/g, "d");
4. ………..………………………….

The **replace** function takes two parameters, first one uses to find character in string you want to convert and The **replace** function takes two parameters, first one uses to find character in string you want to convert and second is the character which you intend to put instead of finding litter (replace). The first argument takes character between two slashed symbols / /, this means, it uses regular expression object to find Kurdish/Arabic letter, and (/ /g) uses to search without stopping until find all matches. If it omits, it will stop in first matching (Stoyan, 2008).

When above rule is applied to all characters (each character has it's against one), the converted script is understandable, but spelling is incorrect (messy), so to reduce spelling errors, cases (irregular cases) will be taken from both origin script and converted script. Look at below table, two words, (دهوك) means Duhok city and (يارى) means game are taken as example of converting each character to its opposite, converted word is misspellings, in this case converted word needs to re-converted, to reduce misspellings. To reduce misspellings.

Table (4) : Two Examples of cases after converting

Cases	Origin word	Characters of word	Converted word (misspellings)	Re-converted word (correct spelling)
Case 1	دهوك	د ه و ك	d h o k	d i h o k
Case 2	يارى	ى ا ر ى	î a r î	y a r î

Case #1

There are characters that have more than one shapes (more than Unicode) with same sound depend in the position in the word, see Table below (Unicode, 2014)**.**

Table (5) : Case #3 of converting (ک, ك) to (k)

Position in word	**Example**	**Unicode**
Begin of word	كوردستان	U+0643
Middle of word	ئاڤاكرن	U+06A9
Last of word	دهوك	U+0643

As shown in the Table above, there are two characters (ک, ك) should be converted into one Latin character (k), this can be done:
1. word = word.replace (/ك/g, "ک"); // this convert ک to ك if is available
2. word = word.replace(/ک/g, "k"); // then convert ك to k in Latin.

Case #2

Character (i) is a vowel character available in Latin scrip but not available in Kurdish/Arabic script (see Table below). This character has a sound in Kurdish but it is not written in Kurdish/Arabic script, so it is difficult to predicate where this character can be written when converting to Latin but we can use it if we found these following converted characters have a space before and after them. See Table below.

Table (6) : Insert (i) after characters b, j, l, ç and d

Converted chars after and before space	**Re-converted**	**Example**	
B	bi		
J	ji	ژ سالا ...	ji sala ....
L	li	ل چیایێ جودى...	li çiyayê cudî
Ç	çi		
D	di		

1. word = word.replace(/ b /g, " bi ");
2. word = word.replace(/ ç /g, " çi ");
3. word = word.replace(/ d /g, " di ");
4. word = word.replace(/ j /g, " ji ");
5. ……………………………………

Also, (i) can be putted between two contiguous characters as shown in Table **Error! Reference source not found.**:

Table (7) : put (i) between two of contiguous characters

Converted contiguous characters	Re-converted to	
Bt	Bit	
Bx	bix	
Ch	cih	
..	…	…

1. word = word.replace(/bt/g, "bit");
2. word = word.replace(/bx/g, "bix");
3. word = word.replace(/ch/g, "cih");
4. word = word.replace(/cv/g, "civ");
5. word = word.replace(/dd/g, "did");
6. ………………………………….

Case #3

The character (î) changes before or after characters (a, e, ê, and u) to (y). In addition, it changes to (y) in the beginning of the word (see line 2).

1. word = word.replace(/ی/g, "î");
2. word = word.replace(/ î/g, " y"); //changes (î) to (y) in the beginning of the word.
3. word = word.replace(/îa/g, "ya");
4. word = word.replace(/îe/g, "ye");
5. word = word.replace(/îê/g, "yê");
6. word = word.replace(/îo/g, "yo");
7. word = word.replace(/îu/g, "yu");
8. …………………………………

Same case for character (u), it changes to (w) before and after (a, e, ê, o and î)

1. word = word.replace(/ua/g, "wa");
2. word = word.replace(/ue/g, "we");
3. word = word.replace(/uê/g, "wê");
4. word = word.replace(/uî/g, "wî");
5. ………………………

Case #4 (Exceptions)

Spelling of some converted words are not correct due to of character (i), therefore, these words convert again. It helps to reduce spelling errors in algorithm, this case called (exceptions word). Table **Error! Reference source not found.** shows exception words.

Table (8) : Exception case

Word converted	Word re-converted
grtn	Girtin
krn	Kirin
karb	Karib
bgr	Bigir

1. word = word.replace(/grtn/g, "girtin");
2. word = word.replace(/krn/g, "kirin");
3. word = word.replace(/karb/g, "karib");
4. word = word.replace(/bgr/g, "bigir");
5. word = word.replace(/grn/g, "girin");
6. ……………………………………

Converting Latin Characters to Kurdish/Arabic Characters

Latin characters have both capital and small case, so for easy comparison, capital case is converted to small by using toLowerCase JavaScript function, and then lower case is used for comparison. It is easy to convert characters have against characters, as shown in Table **Error! Reference source not found.** but as Latin converting, there are cases which are not follow this rule.

Case #1

As it mentioned character (i) is exist in Latin and is not in Kurdish/Arabic script, so below is a code which is used to handle this:

1. word = word.replace(/ i/g, " ئ"); // beginning of word example istabull -> ئیستەنبول
2. word = word.replace(/i/g, ""); //replaced (i) with nothing.

The above code replaces "ئ" character with "i" if it is in beginning of the word for instance (Istanbul ->ئیستەنبول), otherwise, it replaces with nothing due to "i" is not available in Kurdish- Arabic character

Case #2

Word (e ,a) are equivalence to "ئه ، ئا" in Kurdish- Arabic, but this case is correct in the beginning of words, so if converted is not in the beginning of word the spelling will not be correct . Let take the following as example. "**beraz**" word is a (pig) in English will changes to "**بئەرئاز** **"**, this case "ئ" must be removed from word as follow.

1. word=word.replace(/بئ/g, "ب"); //e-->ئه ---> بئەراز
2. word=word.replace(/پئ/g, "پ");
3. word=word.replace(/تئ/g, "ت");
4. word=word.replace(/جئ/g, "ج");
5. word=word.replace(/چئ/g, "چ");
6. word=word.replace(/خئ/g, "خ");
7. word=word.replace(/دئ/g, "د");
8. word=word.replace(/رئ/g, "ر");
9. ………………………………….

Look at, "ئ" character is removed from "**ئه**", "رئ" by lines number 1and 5.

The full code for both converting can be seen in appendix. The following is demonstrated how we can apply this algorithm for our real application for both programmers (developers) and non-programmer (users).

Table (9) : Latin characters against Kurdish/Arabic characters

b	d	s	F
ب	د	س	ف

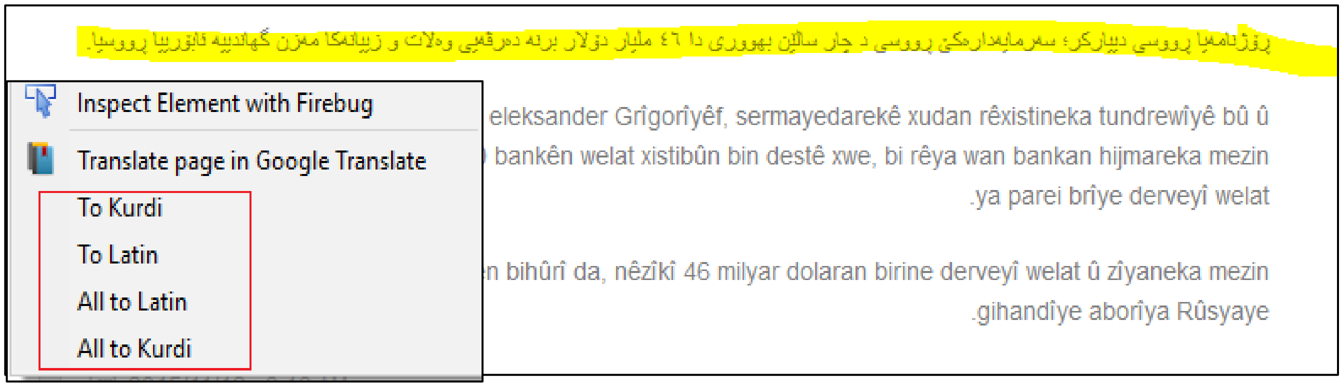
4.Applications Using Algorithm

In order to implement algorithm, two applications create from it, Firefox add-ons and jQuery plugin.

5.FIREFOX ADD-ONS (KURDI_LATIN 0.1)

This Firefox add-ons (extension) is designed to help web users to convert script from one written script to its opposite. For instance, if web site is displayed in Kurdish/Arabic characters, users can convert it to Latin and vice versa, the screenshot bellow shows how one line of **WAAR** Latin script converted to Kurdish/Arabic script. When right-click is fired in Firefox document, context menu is appeared (see Fig () with four options to convert document, the first two options (To Kurdi, To Latin) convert selected written script and others (All to Kurdi, All to Latin) convert whole document.

The Firefox add-ons is authorized with authentication credential, it means anyone can integrated with its Firefox but after that it requires code to activate it. the following page is appeared when first time add-ons installed and the view code should be sent to email (see Fig (), then code will be added to user record to activate for the next time.



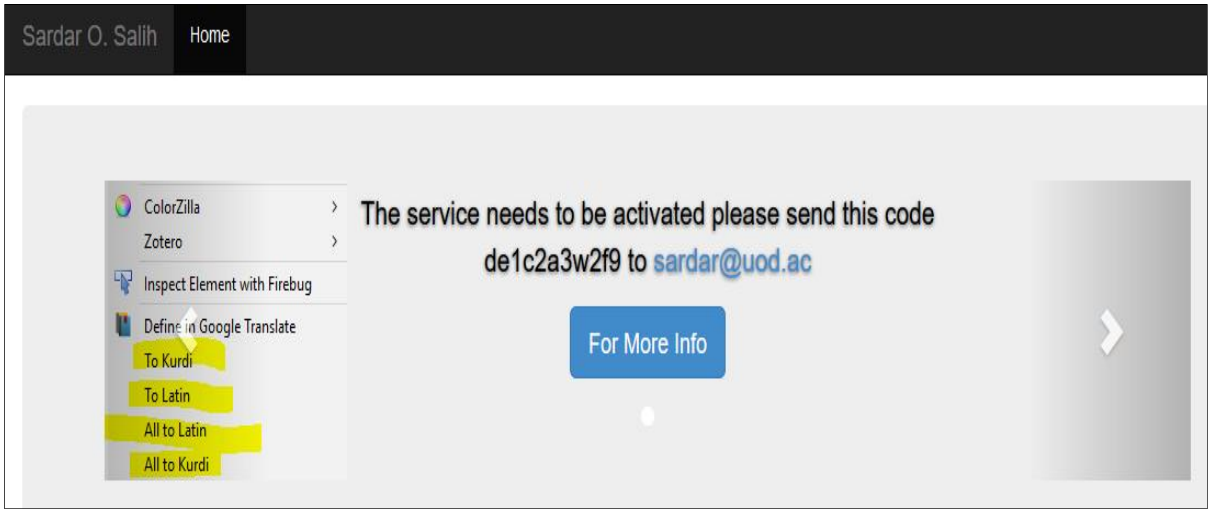
رۆژنامەیا ڕووسی دیارکر؛ سەرمایەدارەکێ ڕووسی د چار ساڵێن بهووری دا ٤٦ ملیار دۆلار برنە دەرڤەیی وەلات و زیانەکا مەزن گەهاندییە ئابۆریا ڕووسیا.

eleksander Grîgorîyêf, sermayedarekê xudan rêxistineka tundrewîyê bû û
bankên welat xistibûn bin destê xwe, bi rêya wan bankan hijmareka mezin
.ya parei brîye derveyî welat

n bihûrî da, nêzîkî 46 milyar doleran birine derveyî welat û zîyaneka mezin
.gihandîye aborîya Rûsyaye

Inspect Element with Firebug
Translate page in Google Translate
To Kurdi
To Latin
All to Latin
All to Kurdi

Fig (1) : screenshot of converted written style (WAAR article)



The service needs to be activated please send this code de1c2a3w2f9 to sardar@uod.ac

Fig (2) : screenshot appears while first time installing

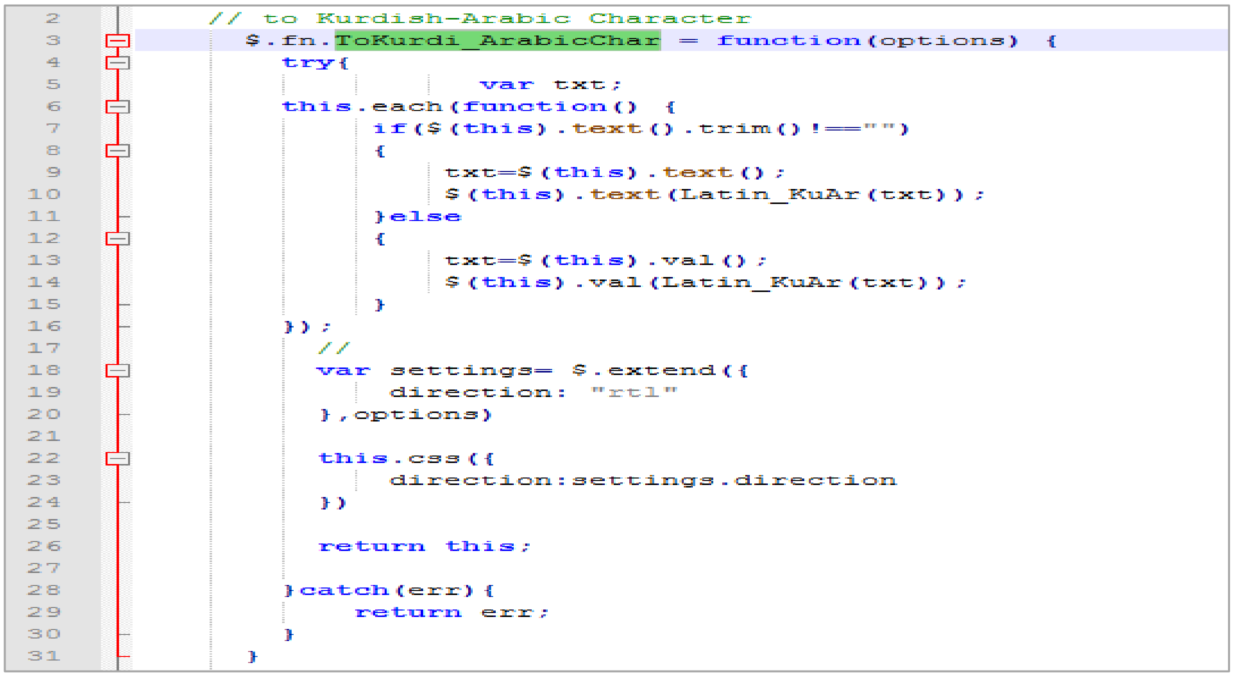
6.Convert Ku-Ws (Convert Kurdish Written Script) Jquery Plugin

jQuery is a JavaScript library which is coded to manipulating DOM (Document Object Model). This library helps developers to increases productivity of its work and reducing its time and cost. **jQuery plugin** is extended to jQuery library which helps developer to inherit all jQuery functionality. In our case "Convert**Ku-WS**" (Convert Kurdish Written Script) plugin is configured on jQuery plugin to help web developers to use in their website. This plugin helps web developers to convert any written script to its opposite.

DESIGNING CONVERTKU-WS PLUGIN

To create jQuery plugin, just add API (algorithm functions) to jQuery $.fn object, this object inherits jQuery library to algorithm API. The following simple example,

explains how jQuery plugin is created (Schlegel, 2014).
$.fn. greenMe = function() {
this.css ("color", "green");
};
$("a"). greenMe (); // Makes all the links green
The greenMe function changes font color of all links in the web document, so, according to this principle, functions (ToKurdi_ArabicChar, ToLatinChar) in algorithm can be configured with JQuery library. The following is to Kurdish/Arabic function which developers can use in their code to convert any HTML text in Latin to Kurdish/Arabic.



```
    // to Kurdish-Arabic Character
    $.fn.ToKurdi_ArabicChar = function(options) {
      try{
            var txt;
      this.each(function() {
          if($(this).text().trim()!=="")
          {
              txt=$(this).text();
              $(this).text(Latin_KuAr(txt));
          }else
          {
              txt=$(this).val();
              $(this).val(Latin_KuAr(txt));
          }
      });
        //
        var settings= $.extend({
            direction: "rtl"
        },options)

        this.css({
            direction:settings.direction
        })

        return this;

      }catch(err){
          return err;
      }
    }
```

Fig (3) : Latin to Kurdish - Arabic function

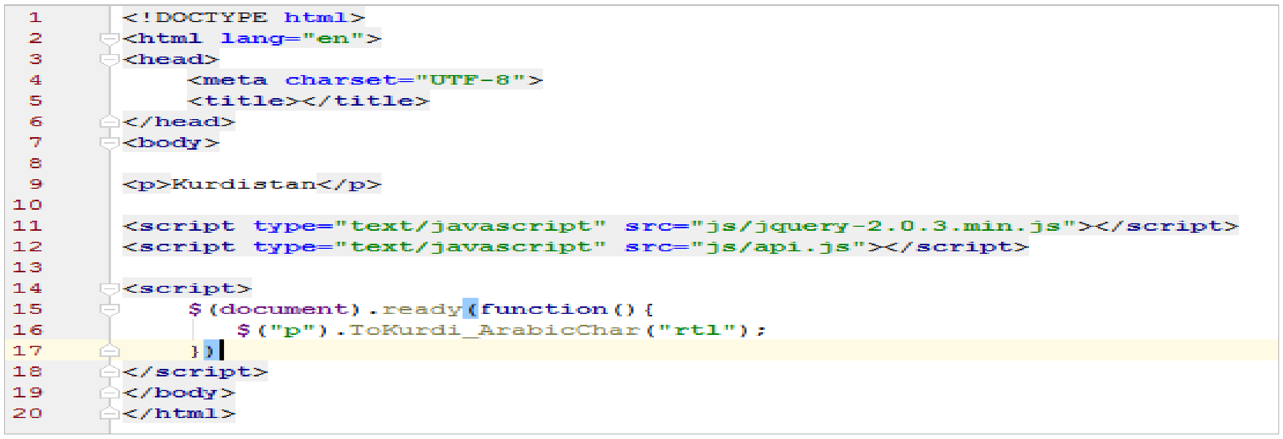
The following demonstrate how developer can convert HTML Latin text to Kurdish/Arabic. To do that developers need to add the following statement.
$(selector). ToKurdi_ArabicChar (options);
Selectors "are patterns used to select the element(s) you want to style", and option is used to specify text direction (left to right and vice versa).

Example #1

If we have this HTML element:
<p> Kurdistan </p>
The P element has a text is written in Latin character to convert it, just put the following statement between **<script>** tag in HTML document:
$(p). ToKurdi_ArabicChar("rtl");
The above statement converts all P text in document to Kurdish/Arabic written script. So, in this case <p> Kurdistan </p> will change to <p>کوردستان</p>! the argument "rtl" changes element written direction from left- right to right-left due to Kurdish/Arabic written script is right to left.

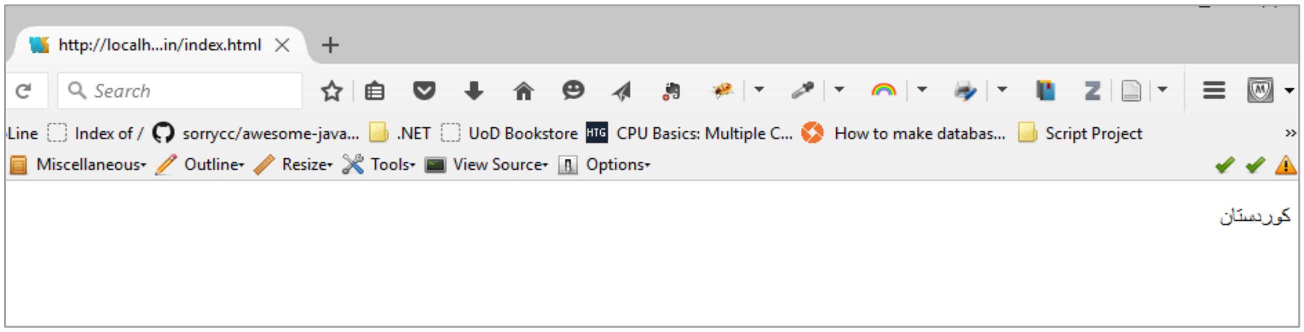
Full Code

The figures 4 and 5 illustrate full code of converting Latin to Kurdish/Arabic characters.



```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title></title>
</head>
<body>

<p>Kurdistan</p>

<script type="text/javascript" src="js/jquery-2.0.3.min.js"></script>
<script type="text/javascript" src="js/api.js"></script>

<script>
        $(document).ready(function(){
            $("p").ToKurdi_ArabicChar("rtl");
        })
</script>
</body>
</html>
```

Fig (4) : Full code of converting P elements text in Latin to Kurdish- Arabic script



کوردستان

Fig (5) : Result of converted text in p element(s)

The result of above code is illustrated in the following figure.

Example #2

If developer has this HTML element:

<p> کوردستان</p>

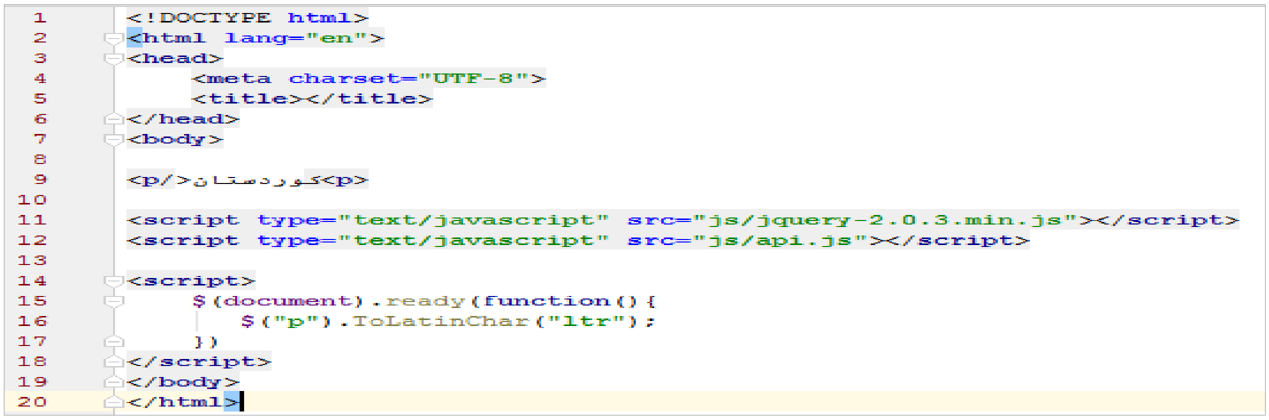
To convert it to Latin, just write down the following code.

$(p). ToLatinChar ("ltr");

The above statement converts all <p> texts in document to Latin written script, so, in this case <p> کوردستان </p> will change to <p> Kurdistan</p>! The argument "ltr" changes element text direction from right-left to left-right due to Latin written script is left to right.

The direction argument is optional, if it is omitted, it will change direction automatically according to written script direction, Kurdish/Arabic (right-left) and Latin (Left-right).

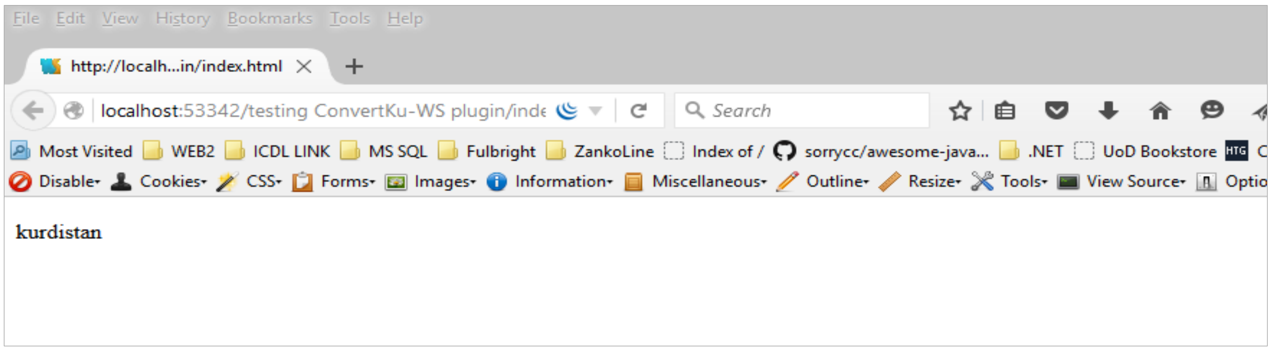
The figures 6 and 7 illustrate full code of converting Latin to Kurdish/Arabic characters.



```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <title></title>
</head>
<body>

<p>کوردستان</p>

<script type="text/javascript" src="js/jquery-2.0.3.min.js"></script>
<script type="text/javascript" src="js/api.js"></script>

<script>
        $(document).ready(function(){
            $("p").ToLatinChar("ltr");
        })
</script>
</body>
</html>
```

Fig (6) : Full code of converting P element text in Kurdish- Arabic to Latin script

The result of above code illustrates in the following figure.



kurdistan

Fig (7) : Result of converted text in p element(s)

As you saw, this plugin changes written text in document from one script to its opposite programmatically without need to write text in both written scripts for two versions, this helps developers to design one version of Kurdish written script in their website, instead of two. The following benefits of designing one version of website instead of two are:

1. Reducing time and cost for both web developers and data entry.
2. Don't need to have two site administrations.
3. Writing text in one version, not need to have two versions (Kurdish/Arabic and Latin).

7.CASE STUDY

WAAR media website is taken for case study to find how algorithm and its application Firefox add-ons can convert

written script from one script to another, WAAR has two version of written script (Latin and Kurdish/Arabic).
The bellow is a part of WAAR article which is written in Latin with its converted version using Firefox add-ons.

Rizgarkirina Şingalê bi serpereştîya Barzanî rojeka dîrokî bû
Mêjû: 2015/11/17 - 2:14 PM
Waar –Duhok:
Mîr Tehsîn Beg Mîrê Êzîdîyan, rizgarkirina Şingalê ji alîyê hêzên pêşmerge ve bi serpereştîya serokê herêma Kurdistanê, bi rojeka dîrokî bi navkir û ragîhand; herçend Kurdên Êzîdî qurbanîyên yekê yên êrîşa ser Şingalê bûn, lê niha hind alî ketine çandina tovê ajawegêrîyê dinava xelkê Şingalê da.

ڕزگارکرنا شنگالێ ب سەرپەرشتییا بارزانی ڕۆژەکا دیرۆکی بوو
مێژوو: ١٧/١١/٢٠١٥ - ٢:١٤ پم
وائار –دوهۆک:
میر تەهسین بەگ میرێ ئێزیدییان، ڕزگارکرنا شنگالێ ژ ئالییێ هێزێن پێشمەرگە ڤە ب سەرپەرشتییا سەرۆکێ هەرێما کوردستانێ، ب ڕۆژەکا دیرۆکی ب ناڤکر و ڕاگیهاند؛ هەرچەند کوردێن ئێزیدی قوربانییێن یەکێ یێن ئێریشا سەر شنگالێ بوون، لێ نها هند ئالی کەتنە چاندنا تۆڤێ ئاژاوەگێرییێ دناڤا خەلکێ شنگالێ دا.

Below is a piece of sample Kurdish/Arabic article in WAAR media and its converted version using algorithm with Firefox add-ons.
If you notice, that both converted scripts give clear text and they can be readable for users. But if you look at word 'dhuk' in **Error! Reference source not found.**, it gives you clear meaning but the spelling is not correct because (i) letter is missed, the correct is like 'Dihuk'. This occurs due to letter (i) is not written in Kurdish/Arabic characters. But if you look at **Error! Reference source not found.** again, you will see word (bi) is appear in some case, this is due to it follows rule in case #3 Kurdish/Arabic to Latin. This case study shows that the algorithm and its Firefox add-ons can give clear text while it is doing converting, but it rarely gives spelling error.

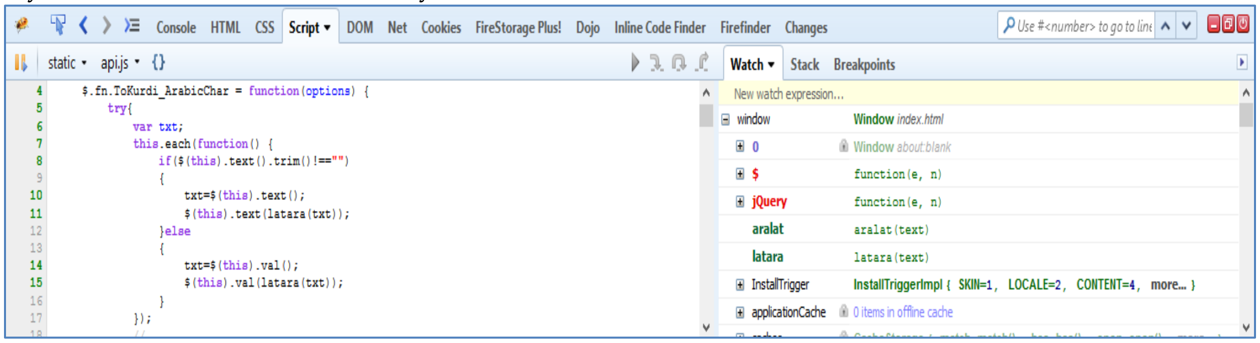
7.1 TESTING

There are two tests can be conduct to find performance of algorithm, the first one, is conducted by (developers) which have knowledge about the structure of software, this type of testing is call white-box testing and the second one, is called black-box testing, this testing is done by who don't have acknowledgment about the software (Laurie, 2011).

7.2 WHITE-BOX TESTING

White box testing focuses on the internal structure of the software code (Khan, Khan, & others, 2012). This test is conducted by system developers using debugging tool such as Firebug, also, to avoid system from crashing. JQuery plugin and algorithm API are wrapped with try-catch errors exception, the try-catch helps to find errors.

```
try
{ // code here
[throw(exception)] }
catch(err)
{//Handle errors here }
```



```
    $.fn.ToKurdi_ArabicChar = function(options) {
        try{
            var txt;
            this.each(function() {
                if($(this).text().trim()!=="")
                {
                    txt=$(this).text();
                    $(this).text(latara(txt));
                }else
                {
                    txt=$(this).val();
                    $(this).val(latara(txt));
                }
            });
```

```
window          Window index.html
  0             Window about:blank
  $             function(e, n)
  jQuery        function(e, n)
  aralat        aralat(text)
  latara        latara(text)
  InstallTrigger   InstallTriggerImpl { SKIN=1, LOCALE=2, CONTENT=4, more... }
  applicationCache 0 items in offline cache
```

Fig (8) : screenshot of firebug use

7.3 BLACK-BOX TESTING

This test is conduct to find errors and accuracy of the software. Five random testers are selected for this test (JAKOB NIELSEN, 2000). These users will be asked to convert one article of five common media websites in Kurdistan for both script, after converting, they will be asked their feedback to analyze and to check algorithm works as it is supposed to do, and how users satisfy, as well as, to determine how the algorithm is accurate (less spelling errors) while converting text.

7.4 USERS FEEDBACK

As mentioned, testers will be asked to install Firefox add-ons and open random article in Kurdish (both written script) in one websites (Rudaw, WAAR, NRT, K24 and Ronahi), then they will do converting by using Firefox add-ons, after that, they will check converting text and read carefully with answers following questions with its satisfaction to check accuracy.

Table (9) : Testers opinion for accuracy of algorithm

#	Statements	User1	User2	User3	User4	User5

1	Converted text is understandable	Sa	Sa	Sa	Sa	Sa
2	Spelling of converted text is well	A	A	Sa	A	Sa
3	In general, converted text is useful for you.	Sa	Sa	Sa	Sa	Sa
4	These add-ons help you if you have not understood text written in one script.	Sa	Sa	Sa	Sa	Sa
5	You can depend on the converted text for your official document.	A	SA	A	SA	A

(Sa) Strongly Agree (A) Agree (D) Disagree (Sd) Strongly Disagree

These above statements are answered in the web-based application, then the result will be collect in the email. The screenshots of this web based application are appended to appendix.

7.5 TESTING EVALUATION

User feedback test gives how is algorithm performance? White box test shows that the internal of system is working what is supposed to do. This test is conducted by using Firebug debugging which it uses to debug error. Furthermore, try-catch exception uses to avoid code from crashing at run time and inform exception messages. Black box test is conducted to find accuracy (spelling errors) by using questionnaires. There are statements which use to find accuracy of the algorithm, the users opinions show that the algorithm is enough good, but the statement number (2) in (Table (**9**), two users out of five give "Strong agree" and others "agree", this due to rarely spelling of some converted words are not correct depended in the text intended to be converted, also this led users to answer the statement number (5) in (Table (**9**) with 3 "agree" out of five with two "strong disagree".

7.6 Further Aspect of Using Algorithm

Some further ideas are not being applied, so these ideas will be applying in future of the developing software (algorithm and its applications), as it is seen, tests show that the spelling of some converted words are usually not correct, therefore, the algorithm will be developed to reduce spelling error, this can be done with adding more case if available and more exception words to algorithm. In addition, there will be plugins for variety of technology such as angularJS, PHP, C# …etc., this will help algorithm to be more populated and used in community.

CONCLUSION

As it has been explained in the previous sections , the algorithm passes through software engineering steps, requirements, designing, coding, implementing and testing, in the designing, the characters have their against characters in both scripts , they can be easy converted but there are some cases which are not follow this rule ,these cases numbered with them suggestion solutions, after that there are application are made up from algorithm, these application are Firefox add-one and jQuery plugin and they are useful for both users and web developers to convert written script, in the case study and testing the algorithm show that the converted text is clear and understandable but spelling of some converted words usually are not correct, so in this case the algorithm needs to be developed.

REFERENCES

Hossein Hassani, & Dzejla Medjedovic. (2016). AUTOMATIC KURDISH DIALECTS IDENTIFICATION.

JAKOB NIELSEN. (2000). Why You Only Need to Test with 5 Users. Nielsen Norman Group. Retrieved from https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

Jemal Nebez. (2015). Kurdish Academy of Language [KAL] | Kurdish Academy of Language. Retrieved 15 November 2015, from http://www.kurdishacademy.org/?q=node/1

Khan, M. E., Khan, F., & others. (2012). A comparative study of white box, black box and grey box testing techniques. International Journal of Advanced Computer Sciences and Applications, 3(6), 12–1.

Laurie, W. (2011). A (Partial) Introduction to Software Engineering Practices and Methods. NCSU CSC326.

Mozil. (2010). Add-ons. Retrieved 24 November 2017, from https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons

pellk. (2010). pellk Kurdî Nûs 4.0. Retrieved 26 November 2017, from http://www.transliteration.kpr.eu/ku/en.html

Schlegel, K. (2014). Balloon synopsis: a jQuery plugin to easily integrate the semantic web in a website? In Proceedings of the 2014 International Conference on Developers-Volume 1268 (pp. 19–24). CEUR-WS. org.

Stoyan, S. (2008). Object-Oriented JavaScript. Packet Publishing Ltd.

Unicode, O. (2014). Arabic Range: 0600–06FF. unicode.org.