World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:10, No:6, 2016

Approximately Similarity Measurement of Web Sites Using Genetic Algorithms and Binary Trees

Doru Anastasiu Popescu, Dan Rădulescu

Open Science Index, Computer and Information Engineering Vol:10, No:6, 2016 publications.waset.org/10004699/pdf

***Abstract*—**In this paper, we determine the similarity of two HTML web applications. We are going to use a genetic algorithm in order to determine the most significant web pages of each application (we are not going to use every web page of a site). Using these significant web pages, we will find the similarity value between the two applications. The algorithm is going to be efficient because we are going to use a reduced number of web pages for comparisons but it will return an approximate value of the similarity. The binary trees are used to keep the tags from the significant pages. The algorithm was implemented in Java language.

***Keywords*—**Tag, HTML, web page, genetic algorithm, similarity value, binary tree.

I. Introduction

GENETIC algorithms are used to solve optimization problems. They belong to the class of Evolutionary Algorithms and they can be used in a big variety of fields such as image processing, computational physics, artificial intelligence and even agriculture (as in [4]). The genetic algorithms have been inspired by natural processes and it is considered a heuristic algorithm. Also, there have been developed methods for determining the similarity of two web sites but they are not efficient as they work with a big number of web pages. Examples of such algorithms are written in [5] and [6]. Other applications of the genetic algorithm can be found in [1]-[3]. In [7], it is described a similarity measurement algorithm using genetic algorithms. Papers [8]-[10] show different ways of determining the similarity between two web sites. A very interesting application is using the edit distance, which can be combined with a genetic algorithm as in [11]-[13].

In our paper, the genetic algorithm will return a chromosome for each web application. These chromosomes will be represented by a binary tree which keeps a set of distinct tags from our applications. We will present in detail the genetic elements and the structures used in the algorithm in Section II. We will also give an example to show the exact meaning of the presented definitions. In Section III, we will present the algorithm sequences and in Section IV, we will present the results obtained from testing and we are going to group the data into a table and diagrams. In Section V, we present the conclusions and information about future work and in the last one the references.

Doru Anastasiu Popescu is with the University of Pitesti, Faculty of Mathematics and Computer Science, Romania (e-mail: dopopan@yahoo.com).

Dan Rădulescu is with the Department of Computing, National College "Radu Greceanu", Slatina, Romania (e-mail: dan_radulescu96@yahoo.com).

II. Presenting the Algorithm and Examples

Our algorithm is divided in two important steps. The first step is the one where we determine a significant set of tags for each web site. And in the second step, we determine the similarity value by using a specific method.

Firstly, let's present the genetic elements. We will start with a population of chromosomes containing a certain number of chromosomes; each chromosome contains the same number of genes. Each chromosome will be characterized by a set of web pages (kept by an array) and by a binary tree (formed with the distinct tags from the web pages found in the array presented above). We will have a sequence for mutation one for cross-over and one for selection. Now we are going to present the performance function (also called fitness function). We will have an array in which "element i gives us the number of distinct tags from chromosome i". The best chromosome it is the one with the biggest number of distinct tags. So, at the end of this part we will have determined a binary tree for each web site, containing a set of significant tags. We will use web pages e1, e2, e3 and e4 as examples. We will use inorder traversal to show the binary trees.

The final sentence of a caption must end with a period.

TABLE I
HTML Code for Web Pages e1 and e2

e1.html	e2.html
<HTML>	<HTML>
<HEAD> </HEAD>	<HEAD> </HEAD>
<BODY> example1	<BODY>
<U> a1 </U>	<I> a2 </I>
<I> a2 </I>	<U> a1 </U>
</BODY>	</BODY>
</HTML>	</HTML>

TABLE II
HTML Code for Web Pages e3 and e4

e3.html	e4.HTML
<HTML>	<HTML>
<HEAD> </HEAD>	<HEAD>
<BODY> example3	</HEAD>
a1 	<BODY>
</BODY>	<FONT>example4
</HTML>	</FONT>
	<I><U>a3</I></U>
	</BODY>
	</HTML>

- Binary tree for e1.html: </BODY> </HEAD> </HTML> </I> </U> <BODY> <HEAD> <HTML> <I> <U>
- Binary tree for e2.html: </BODY> </HEAD> </HTML> </I> </U> <BODY>
 <HEAD> <HTML> <I> <U>
- Binary tree for e3.html: </BODY> </h1> </HEAD> <BODY> <h1> <HEAD> <HTML>

Open Science Index, Computer and Information Engineering Vol:10, No:6, 2016 publications.waset.org/10004699/pdf

- Binary tree for e4.html: </BODY> </HEAD> </FONT> </HTML> </I> </U> <BODY> <FONT> <HEAD> <HTML> <I> <U>

Now we observe that we obtained four strings. The longest common substring of two strings (let's consider the one obtained from e1 and the one obtained from e4) is: </BODY> </HEAD> </HTML> </I> </U> <BODY> <HEAD> <HTML> <I> <U>.

If we consider the longest common substring obtained from e2 and e3 we have: </BODY> </HEAD> <BODY> <HEAD> <HTML>

Now, for calculating the similarity we consider the next definition.

We consider S1 the string obtained from the binary tree of the first web site and S2 the string from the binary tree of the second web site. We consider S the common substring. Let L1 be the length of S1, let L2 be the length of S2 and L the length of S.

Definition 1. We consider the similarity value between the two web sites S1 and S2: SV= L/(L1+L2).

For the example presented above (though we have web pages in our example not web sites) SV=0.44 (so 44%).

For the second example presented above (another example on web pages not web sites) we obtain SV=0.32 (so 32%).

III. Applied Genetic Algorithm

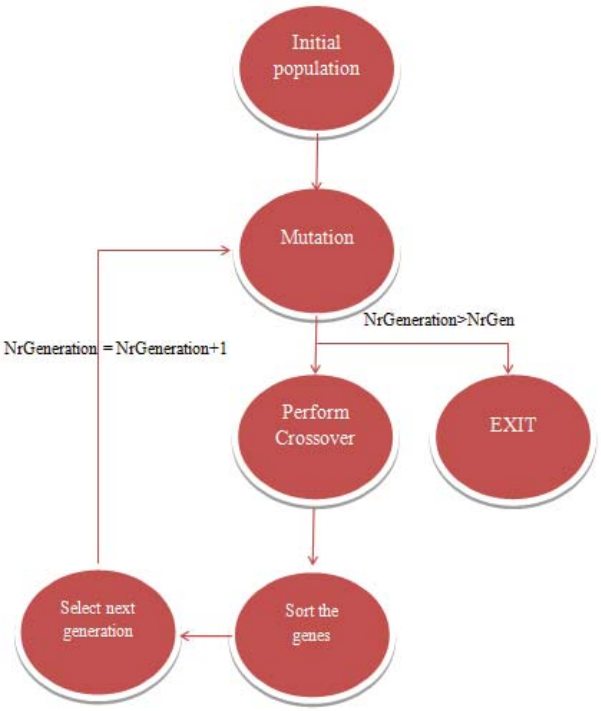


Fig. 1 Scheme of genetic algorithm

We will show the classic scheme of a genetic algorithm, Fig. 1.

- Before the pseudocode we will present the most important variables and the data structures.
- NrMutation = the number of mutations at each step
- NrGene = the number of web pages used for a chromosome (one gene it means one page)
- NrPop = the number of chromosomes for one generation
- NrGenerations = the number of generations
- pages = array with the pages used for each chromosome (it helps when creating the binary tree)
- tree = an array with the binary trees of the chromosomes

We will start with the main sequence of the algorithm:

```
for i=1, NrGenerations do
mutation();
crossover();
sort();
endfor
```

We will apply this sequence for each web site. The mutation and crossover sequences have been presented in [7]. They are classic genetic algorithms operations. In this paper we will focus on the algorithms designed for creating a binary tree and for determining the longest common substring:

```
BINARY_TREE():

for i=1, NrPagesSite do
select a page;
select a tag and add it into the tree;
increase number of tags from the tree;
endfor
for i=1,NrBinaryTrees do
inorder traversal;
determine string;
endfor

LCS():

for i=1, s1.length do
for j=1, s2.length do
      if s1[i]=s2[j] then
         increase length of LCS;
         add last character used
         in another string;
            else
         retain character with the
         biggest chance to obtain
         a longer LCS;
         continue comparisons;
      endif
endfor
endfor
```

The LCS sequence is dynamic programming method (a classic one). The execution time has a polynomial order. The algorithm can return more sets of significant tags (more chromosomes). Using those sets, we can calculate the similarity rank as an average between all the results obtained (we can choose the best two, three or more chromosomes).

The sort() is going to be used for sorting ascending the chromosomes by their number of tags. We will use the BubbleSort method as it follows:

```
for i=1,NrPop do
```

Open Science Index, Computer and Information Engineering Vol:10, No:6, 2016 publications.waset.org/10004699/pdf

```
        for j=i+1,NrPop do
    if (nrtags(chromosome(i))>nrtags(chromosome(j))) then
    swap chromosom(i) with chromosome(j)
    endif
    endfor
    endfor
```

IV. The Implementation

The algorithm was implemented in NotePad, using the Java language. The computer which ran the tests has a 3.10 GHz Processsor and 8.00 GB RAM. We tested the algorithm for different values of NrGenes. We will present the results in one tables and one chart. We are going to use cod source from HTML files from websites [14]-[19], for our examples.

TABLE III
SIMILARITY VALUES DEPENDING ON THE NUMBER OF GENES

Web sites	icor/jatit	sofa2012/jatit
4 genes	0.2929746	0.2688
5 genes	0.3051095	0.2718894
7 genes	0.2906815	0.26300147
8 genes	0.28650904	0.25108853
10 genes	0.2837274	0.26415095

TABLE IV
SIMILARITY VALUES DEPENDING ON THE NUMBER OF GENES

Web sites	sofa2007/icor	sofa2009/sofa2010
4 genes	0.29943502	0.46153846
5 genes	0.2592565	0.46153846
7 genes	0.29553902	0.43822843
8 genes	0.29182157	0.3939394
10 genes	0.29739726	0.45454547

TABLE V
SIMILARITY VALUES DEPENDING ON THE NUMBER OF GENES

Web sites	sofa2007/sofa2005
4 genes	0.40041608
5 genes	0.3963039
7 genes	0.37577003
8 genes	0.37577003
10 genes	0.3778234

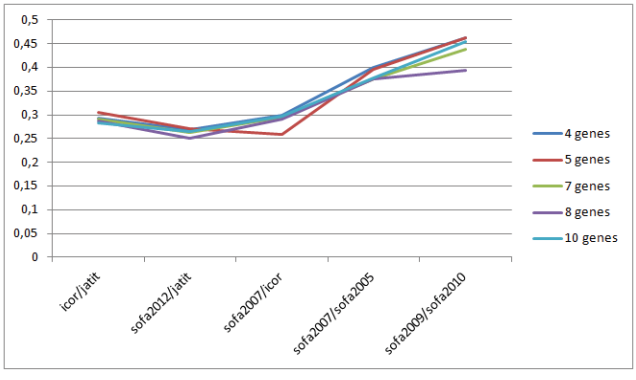


Fig. 2 The results of the similarity value

In Fig. 2, we observe that we obtain similar results when changing the number of genes. The bigger the number of genes is the more accurate the result will be. Of course the execution time will rise proportionally with both the number of genes and the number of generations.

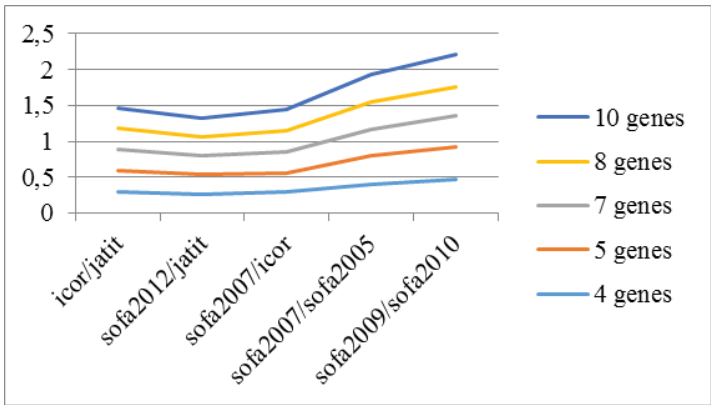


Fig. 3 The execution times of the algorithm when we have different number of genes

In Fig. 3, we observe that the graphics have the same curve no matter how many genes are implied. Certainly, as we have more genes the bigger will be the execution time and this can be seen in the graphic. When constructing a similar chart, but where it depends on the number of generations we obtain a similar figure as the one presented above.

V. Conclusion

In this paper, we presented a genetic algorithm which determines a number of significant tags from a web site. We can use those tags to determine the similarity between two web sites. Using this algorithm, we obtain a good efficiency and good execution times but we obtain approximated results. Our next step is to improve the application on its practical side. We will try to identify exactly the common elements from two web pages with HTML tags using a genetic algorithm. Another interesting objective that we aim is to extend the using of HTML web pages to pages with different language of Internet programming (such as CSS). Genetic algorithms have a wide area of applications and they can deliver interesting results.

References

[1] Koza J.R., Genetic Programming, MIT Press, Cambridge, MA, 1992
[2] N. M. Ciobanu (Iacob), Proposed Algorithm for Solving Queries in a Dynamic System of Distributed Databases, Global Journal on Technology, Vol. 03 pp. 535-540, 2013
[3] C.L Defta, A. Şerb, N.M. Iacob, C. Baron, Threats analysis for E-learning platforms, Knowledge Horizons – Economics, Vol. 6 / Nr. 1, pp. 132–135, 2014
[4] D. A. Popescu, D. Radulescu, Monitoring of irrigation Systems Using Genetic Algorithm, ICMSAO, IEEE Xplore, pp.1-4, 2015.
[5] D. A. Popescu, C. M. Danauta, Similarity Measurement of Web Sites Using Sink Web Pages, 34th International Conference on Telecommunications and Signal Processing, TSP, IEEE Xplore, pp.24-26, 2011.
[6] D. A. Popescu, D. Nicolae, Determining the similarity of two web applications using the edit distance, SOFA, LNCS, 2014, pg.12-20
[7] D. A. Popescu, D. Radulescu, Approximately Similarity Measurement of Web Sites, ICONIP, 2015

[8] Guadalupe J. Torres, Ram B. Basnet, Andrew H. Sung, Srinivas Mukkamala, Bernardete M. Ribeiro, A Similarity Measure for Clustering and Its Applications, ICASA, pp. 1712-1718, 2008.
[9] G. Jeh, J. Windom, SimRank: A measure of Structural-Context Similarity, KDD, ACM, pp. 538-543, 2002.
[10] C. N. Pushpa, J. Thriveni, K. R. Venugopal, L. M. Patnaik, Web Search Engine Based Semantic Similarity Measure Between Words Using Pattern Retrieval Algorithm, CS & IT-CSCP, pp. 1-11, 2013.
[11] P. Zhao, J. Han, Y. Sun, P-Rank: A Comprehensive Structural Similarity Measure over Information Networks, CIKM, ACM, pp. 1-10, 2009.
[12] D. Bollegata, Y. Matsuo, M. Ishizuka, Measuring Semantic Similarity between Words Using Web Search Engines, IW3C2, pp. 757-766, 2007.
[13] D. Lin, An Information-Theoretic Definition of Similarity, ICML, ACM pg. 296-304, 1998.
[14] Journal of Theoretical and Applied Information Technology, http://www.jatit.org (Accessed 10 March 2016)
[15] International SOFA Workshop, http://trivent.hu/2012/ieeesofa2012/ (Accesed 10 March 2016)
[16] International SOFA Workshop, http://trivent.hu/2010/ieeesofa2010/ (Accesed 10 March 2016)
[17] International SOFA Workshop, http://trivent.hu/2009/ieeesofa2010/ (Accesed 10 March 2016)
[18] International SOFA Workshop, http://trivent.hu/2007/ieeesofa2007/ (Accesed 10 March 2016)
[19] International SOFA Workshop, http://trivent.hu/2005/ieeesofa2005/ (Accesed 10 March 2016)

Open Science Index, Computer and Information Engineering Vol:10, No:6, 2016 publications.waset.org/10004699/pdf