Evaluation and implementation of machine learning techniques in usability testing for web sites

Christoffer Korvald, Eunjin Kim, Hassan Reza
Computer Science Department
University of North Dakota
Grand Forks, ND, 58202-9015, U.S.A.
christoffer.korvald@my.und.edu, ejkim@cs.und.edu, reza@cs.und.edu

Abstract

User satisfaction on web sites depends on many factors and usability is one of these factors. A web site should be organized in a logical manner to aid the user in accomplishing their goal. This paper proposes an automated method for usability testing of a web site using machine-learning techniques to uncover usability issues related to the organization of the pages (information architecture). The information architecture was represented as a weighted directed graph. The graph features are used to train several machine learning models that can then be used to predict the usability score of another website. Models evaluated include classifiers of support vector, random forest, decision tree, regression models, etc. A number of machine learning models are evaluated to determine the best possible model for this specific use case, using 10-fold cross validation on various sized datasets. We also demonstrate a way of extracting a ranked list of prominent features to that can be improved.

Introduction

In the market today almost every company and organization has a website. It has become an important source of information and it is still growing at a tremendous rate. Keeping all that information organized and accessible to the end user is a difficult task for the web site owner. With the growing number of users and web sites the competition is also getting harsher and the web site owner will as a result have to work harder to retain users. Wu and Offutt [1] discussed how there appears to be no "site loyalty" for web sites which can make it very hard to hang on to users and when they are likely to move on to another site of better quality. This is one of the reasons why making sure a web site meet user expectations is crucial to the site owner and failure to do so can result in potential loss of revenue.

Making sure the user is satisfied with the web site comes down to many factors and usability is one of them. If the site is perceived by the user to be unusable it will most likely discourage them from coming back. They arrive at the web site with a specific goal in mind and depending on the type of site it might be to purchase goods, gather information, communicate with other users, etc. The web site should be organized in a logical manner to aid the user in accomplishing their goal. Performing testing to uncover potential usability issues is therefore crucial to any web site owner. Usability testing is usually done in two ways (classical and the automated) [2] and will be discussed in further detail later. This paper proposes an automated method to evaluate the usability of a web site using machine-learning techniques. Harty stated that automated usability testing could uncover many types of issues if combinations of several techniques are used [3]. The proposed method is not supposed to be used as the only measure of usability of a website, but should be able to uncover usability issues related to the organization of the pages (information architecture), and can be used in combination with other types of usability testing techniques.

Background

Usability

There are many ways of defining the quality of a web site and one of them focused on in the paper is usability. There are also multiple definitions for usability, but Seffah, Donyaee, Kline and Padda attempted to consolidate this into a single model called Quality in Use Integrated Measurement (QUIM) which includes 10 factors: efficiency, effectiveness, productivity, satisfaction, learnability, safety, trustfulness, accessibility, universality and usefulness [4]. The factors are further broken down into 26 criteria that can be measured. The method proposed in this paper will only focus on a subset of those 10 factors: efficiency, productivity, satisfaction and learnability.

Scholtz discusses the advantages and disadvantages of the different approaches to usability evaluation and the phases of usability engineering, which are requirements analysis, design/testing/development, and installation [5]. The main benefit of a model based evaluation approach is that once the model has been defined it can be used repeatedly without much extra cost, while the disadvantage to this approach is that it is difficult and takes time to define the model for the first time. On the other hand user-centered evaluations are good because they involve actual users and the results can uncover specific aspects of a system that might cause problems for the user. On the other hand this is usually a very time consuming and expensive test to administer. Hwang and Salvendy found that 10 ± 2 is the optimal number of test users needed to discover 80% of usability issues [6]. Scholtz goes on to discuss how expert-based evaluation is usually less time-consuming and less expensive than the user-based approach, but possibly not as accurate as there are fewer individuals reviewing the site and they might not be in the same demographic as the actual users of the web site [5].

Rukshan and Baravalle also discussed the differences between the automated and classic usability evaluation techniques where they show that with automated usability evaluation you could reach a larger number of subjects with a larger geographic demographic and focus on breath compared to depth [2].

Sonderegger and Sauer showed that there are other issues with user-based usability evaluation where the presence of an observer in a laboratory would affect the subject and their emotion, performance and physiological measures [7]. They also go into how the set-up of the laboratory could affect responses from subjects.

Evaluating link structure

Zhou and Chen came up with a method of evaluating the link structure of a web site using information about user behavior [8]. They first define a link structure model, which is represented as a weighted directed graph where the nodes are the individual web pages and the hyperlinks are the edges. Next the edge weights are calculated based on the user behavior that is extracted from web logs. When the edge weighted directed graph is created they could calculate the web site complexity using Association Degree and Convenience Degree of page pairs. Kung, Liu and Hsia also create a model from the link structure, but instead of a directed graph they create a Page Navigation Diagram, which is a finite state machine (FSM) [9].

Jacko and Salvendy explored how breath and depth of a menu design influence task complexity [10]. They specifically did a study on the following menu structures: 2^2, 2^3, 2^6, 8^2, 8^3 and 8^6 (X^Y where X is the number of choices at each level and Y is the number of levels). The results showed that as the menu depth increased so did the perceived complexity of the task. Campbell discussed 4 characteristics that describe complexity: multiple paths, multiple outcomes, conflicting interdependence among paths and uncertain/probabilistic linkages [11].

Machine Learning in usability testing

Machine learning has been used in usability testing before and a great example of that is the work of Oztekin, Delen, Turkyilmaz and Zaim where they compare four different models (multiple linear regression, decision trees, neural networks, and support vector machines) and how they performed on the data collected [12]. The data was collected using a UseLearn checklist focusing on factors such as error prevention, visibility, flexibility, accessibility, etc. from the users of an online cell biology course. With 10-fold cross-validation they trained and analyzed the results from the different models. With the data they were able to identify that the multi-layer perceptron neural network performed better than the others with their data. The ability to identify the important features that had the biggest impact on the overall usability score was also discussed.

Approach

The method proposed is very similar to the approach taken by Oztekin, Delen, Turkyilmaz and Zaim [12]. Instead of using data collected from a questionnaire the data used to train the models will be the characteristics of the navigation graph / web site graph and also consider different machine learning models.

The steps of the proposed method are as follows:

1. A simple breath-first search technique to traverse and crawl a web site.
2. Create a directed graph from the data gathered by the crawler
3. Get characteristics of the graph structure
4. Train machine learning models
5. Determine the best model to be used
6. Rank the features to determine which aspects of the web site graph has the biggest impact on overall usability
7. Apply new data to the trained models to predict the usability of new web sites

Web crawler

Web sites are made up pages using a multitude of technologies/languages such as HTML (Hypertext Markup Language) for structuring the document, CSS (Cascading Style Sheets) for the look and feel and some times JavaScript for frontend logic. A backend programming/scripting language is commonly used for dynamic content generation. This paper will focus mostly on small to medium web sites with relatively static content that does not rely heavily on frontend technologies such as JavaScript. How the pages are liked together and organized can be referred to as the web site graph or information architecture and is what will be used to determine the level of usability of a the web site. The ability to crawl the entire web site depends on a number of factors like forms (where the input given might lead to different pages), client-side validation and server-side manipulation as discussed by Marchetto,Tiella, Tonella, Alshahwan and Harman [13].

The simple crawler that was created receives a domain (e.g. http://python.org) to keep the crawling contained within the domain and a start URL (e.g. http://python.org/about/) to indicate where to start the crawling. It is preferred to crawl the entire web site, but in very large sites this can be very time consuming. In a web site graph with N number of nodes where every page is linking to every other page and to itself there will be N^2 number of edges.

There has been work done by Liu, Janssen and Milios on creating smarter crawler based on using user data and Hidden Markov Models (HMM) to crawl the most relevant pages first [14], but since we need the entire web site graph it doesn't matter what order they are crawled as long as all pages are reached. A simple Breadth-First Search approach was taken for simplicity. Time to crawl the entire website will depend on bandwidth and number of pages and hyperlinks. The results are stored in a database to allow the next steps to work with the data without having to wait for the crawler to run on the web site every time.

Graph generation

The graph is then created from the data gathered during crawling with first adding all the pages as nodes N. After all the pages are added to the graph, edges E can be added to represent the link from node i to node j. An example representation of a graph created using this technique in Figure 1. Using a similar technique as described by Zhou and Chen [8] weights could be added to the edges for further analysis, but that is outside of the scope of this paper.

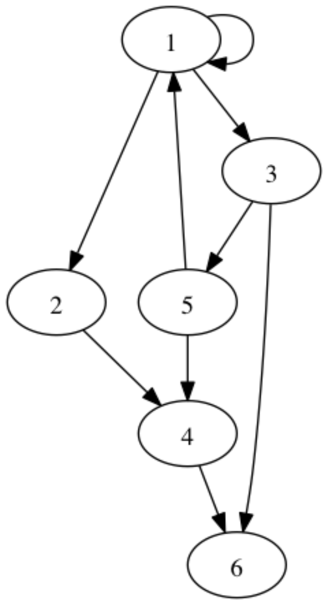

Figure. 1. Example graph created by the algorithm.

The directed graph is defined as follows:

$$G = (N, E)$$

$$N = \{N_i : i \in \{1, n\}\}$$

$$E = \{E_{i,j} : i, j \in \{1, n\}\}$$

From this web site graph the data used by the machine learning models will be extracted. The five features used in this method area as follows, but could be expanded in future studies:

1. Number of nodes (pages)

2. Number of edges (unique links between pages)
3. Average out degree
4. Average in degree
5. Graph radius (the minimum eccentricity of the graph)

$$numNodes = |N|$$

$$numEdges = |E|$$

$$avgOutDegree = \frac{1}{|N|}\sum_{n \in N} \text{outDegree}(n)$$

$$avgInDegree = \frac{1}{|N|}\sum_{n \in N} \text{inDegree}(n)$$

Training data generation

The machine learning models require training data to be able to make predictions. To be able to experiment with different training data set size a program was created to generate semi-random data. The program would take as input the filename where the data should be saved and the number of lines to generate. The following features were favored in order to force certain features to matter more in order to create more realistic user data: graphRadius, numNodes, avgOutDegree and avgInDegree. Example lines from the CSV (comma separated value) training data file in Table 1 where the first column corresponds to 1 = good usability, 0 = bad usability and the rest of the columns corresponds to the list of 5 features provided earlier.

1	545	283803	20	65	2
0	167	9863	92	50	4

Table. 1. Example training data file

Machine learning models

The machine learning models compared are a subset of models provided by the Python machine learning library called scikit-learn [15]. The following classifiers and regression models are compared (named like they are in the scikit-learn library): LinearSVC, SVR, SVC, GaussianNB, RandomForestClassifier, GradientBoostingClassifier, DecisionTreeClassifier, KNeighborsClassifier, ElasticNet, LinearRegression, LassoLars, BayesianRidge, PassiveAggressiveRegressor, PassiveAggressiveClassifier.

A wide variety of models were chosen to compare their performance under different circumstances with the data generated in the previous step. All of the models have revealed different strengths and weaknesses in the results. Some require larger data sets to adequately train, while others can perform reasonably well with the smaller data set.

Evaluating the models

To determine the best machine learning model for evaluating the usability of a website, we need a numerical scheme that evaluates each model with a numeric score. The evaluation of the models was done using k-Fold cross-validation. Kohavi explains that k-Fold Cross-validation is a technique where the dataset is divided up into k mutually exclusive sets which is used to train and test the model's accuracy [16]. It will take one of the newly created sub sets and train the model and then take another sub set to validate the results. Kohavi also showed in his study that a good value k-Fold cross-validation would be k=10. 10-fold cross validation was used in this experiment and produced good results with the different data set sizes.

Each prediction by the model has to be scored and a popular scoring function is log loss, which has been used by many including Roy and McCallum [17] in their work on optimal active learning through Monte Carlo Estimation of Error Reduction. Log loss is a function where X is the actual values and Y is the predicted values:

$$logloss(X,Y) = -\frac{1}{|X|}\sum_{i=0}^{|X|} X_i \ \log Y_i + (1 - X_i) \ \log(1 - Y_i)$$

$$Eval(model) = \frac{1}{N}\sum_{i=1}^{N} logloss(model_i.actual, model_i.predictions)$$

$$where\ N = number\ of\ sets$$

The overall score for the model was computed by taking the mean log loss for each of the cross validation sets. This procedure was repeated with test data sets of size 50, 100, 500, 1000, 2000 and 5000, respectively, to evaluate the performance and results from the different models. Some of the models might work better with a large amount of data, while other can make about the same prediction with a smaller data set. It would be preferred to identify a model that performs adequately with a small data set, as this would reduce the amount of data gathering needed for a final implementation of this method. Getting 5000 users to rate websites will be very time consuming, but 50-100 is attainable in relatively short time.

Important features

For convenience the Random Forest Classifier in the Scikit learn machine learning library [15] was used to extract the important features from the model. After training the model the features can be extracted and sorted by importance to give a list of the features that has the biggest impact on the end result. Standard deviation is also calculated and displayed as a blue line on a graph giving a good overview of which features are important. In this example you can see that features 4 and 3 are very important and feature 2 only has about 6% importance. This shows that the model was able to determine the important features from the dataset described earlier.

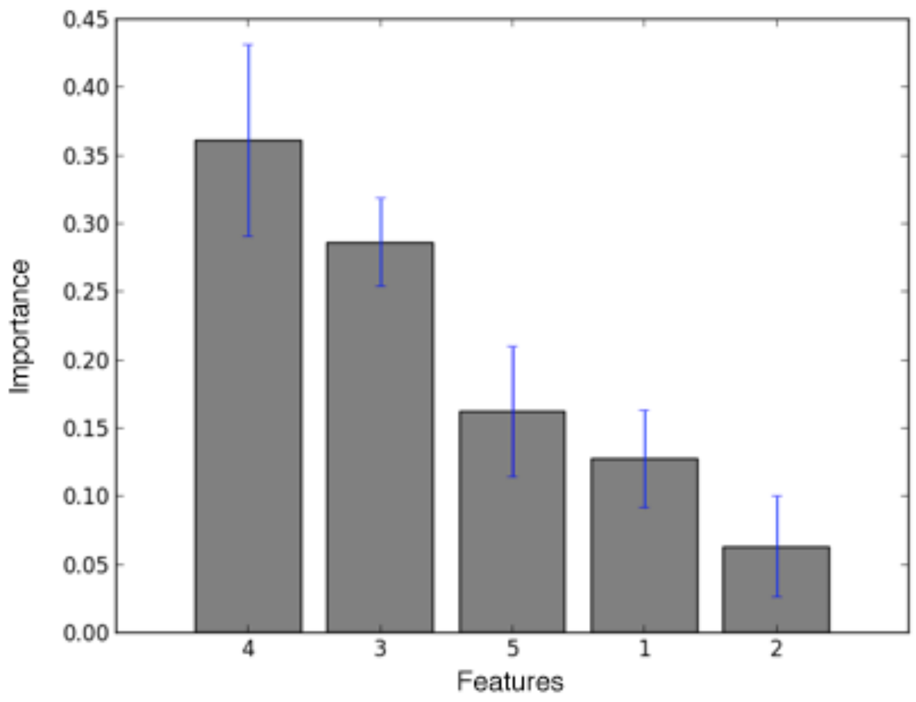

Figure. 2. Example of a graph showing the important features

This is very valuable information for a web site owner looking to improve the usability of a web site as they can focus their work on the most important features that will give a better usability score and as a result give the users a better experience. Work improving feature 2 would not be recommended, as it will not have a great impact on the end result significantly and as making changes might be very time consuming and expensive it is up to the web site owner to decide if it is worth the small improvement in the overall usability.

Results

The crawler was tested on a small web site running locally on the machine as well as a similar web site not running locally. It would identify all the links from every page crawled and store the structure to a database for use in the web site graph generation. It also reduced the amount of information stored by only considering unique links and adding up duplicates and storing that in the database as well.

A custom directed graph class was written in python and it could populate the nodes and edges using the information from the database. As performance has not been optimized with this class it tends to run slower than originally expected. It would be preferred to move this codebase over to a third party library that has been optimized for performance.

Training data sets were created with different sizes to evaluate the impact this would have on the models ability to predict the correct answer. Results shown in Table 2, Figure 2 and Figure 3. With a dataset fewer than 500 the results showed that 11 out of the 14 models performed rather poorly, but 3 performed reasonably well (BayesianRidge, LassoLars and Linear Regression). As the data set reached 500, most of the models started getting closer and at 5000 most of the models performed at a reasonable level. The 3 models that had the worst performance with both small and large data sets were PassiveAgressiveClassifier, PassiveAgressiveRegressor and LinearSVC. It would not be recommended to use these as they require large data sets. It is however interesting to see that LinearSVC performed so badly while another model in the same family, SVR, performed really well even with small data sets. The most consistent model was LinearRegression with a standard deviation of 0.02826 and performed best with data sets 50 and 100. This would be the preferred model when the size of data sets is low, but it also performed very well with a larger data set.

The identification of important features using a Random Forest Classifier was also successful and could be used to suggest to a site owner which aspects of the information architecture to improve first. Just simply giving a score of good/bad for the web site as a whole is not as useful as pointing to specific issues that can be fixed to improve overall usability.

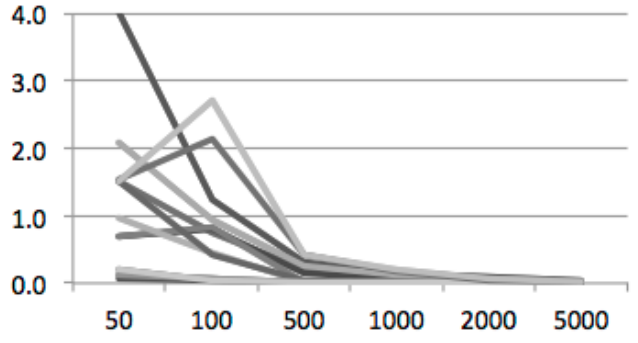

Figure. 3. Scoring of all models. (Dataset size: 50,100,500,1000,2000,5000)

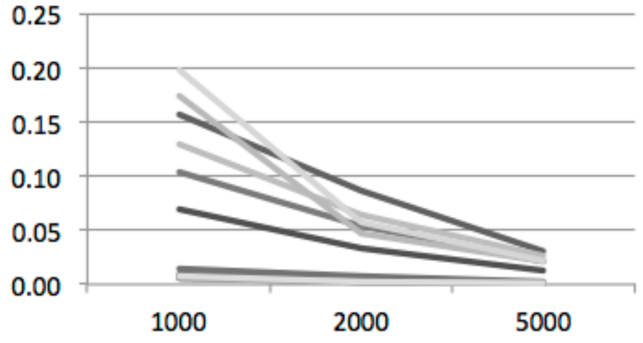

Figure. 4. Scoring of all models. (Dataset size: 1000,2000,5000)

Table. 2. Scoring of models

	50	100	500	1000	2000	5000	STD	AVG
LinearSVC	4.00652	1.24341	0.32190	0.15750	0.08669	0.03068	1.41644	0.97445
SVR	0.11036	0.05480	0.01222	0.00616	0.00310	0.00124	0.03979	0.03131
SVC	1.51974	0.75987	0.20309	0.10396	0.05319	0.02135	0.54191	0.44353

GaussianNB	0.69079	0.79441	0.13678	0.06908	0.03385	0.01256	0.32402	0.28958
RandomForestClassifier	0.69079	0.82894	0.02211	0.00967	0.00397	0.00122	0.35615	0.25945
GradientBoostingClassifier	0.96710	0.44901	0.01658	0.00587	0.00345	0.00108	0.36289	0.24052
DecisionTreeClassifier	1.51972	0.41447	0.03316	0.01485	0.00708	0.00254	0.55098	0.33197
KNeighborsClassifier	2.07236	0.93256	0.25559	0.12849	0.06476	0.02604	0.73404	0.57997
ElasticNet	0.20171	0.05256	0.01160	0.00598	0.00264	0.00116	0.07184	0.04594
LinearRegression	0.07836	0.04749	0.01588	0.00725	0.00310	0.00128	0.02826	0.02556
LassoLars	0.11037	0.05461	0.01215	0.00612	0.00309	0.00124	0.03978	0.03126
BayesianRidge	0.20121	0.05291	0.01593	0.00726	0.00303	0.00128	0.07116	0.04694
PassiveAggressiveRegressor	1.53266	2.14925	0.41029	0.17364	0.04713	0.02098	0.82046	0.72233
PassiveAggressiveClassifier	1.51974	2.69403	0.41447	0.19825	0.05742	0.02190	0.98074	0.81764
Max	4.00652	2.69403	0.41447	0.19825	0.08669	0.03068	1.41644	0.97445
Min	0.07836	0.04749	0.01160	0.00587	0.00264	0.00108	0.02826	0.02556

Conclusion

An automated usability testing method is proposed and different machine learning models evaluated and compared. The comparison successfully identified good (and bad) models for use in situations with smaller and larger data sets. The result may vary when real world data is used, but the 10-fold cross-validation technique was still successful in evaluating the different models used.

The method also demonstrated a way to extract the important features the Random Forest Tree Classifier used and give the web site owner a ranked list of which features to focus on when trying to improve the usability.

Future Work

There is still much to work that could be done with giving useful feedback to the web site owner. The first thing would be to expand the number of features extracted from the web site graph to be used by the models to provide more accurate results.

Different features might be important in different web site categories like an online bookstore compared to a personal blog. It would be interesting to see if different categories favored the same features. Training the models with data for a specified category and comparing the results from the other categories would be able to uncover any differences.

At the current state of the implementation the data sets used to train the models are randomly generated, but in the future it would be interesting to use actual user data. It would not be particularly hard to gather this data as all that is needed is the web site graph, which is automatically generated by the crawler, and ask the user how they would rate the web site (good or bad usability). This process could even be done remotely without bringing subjects into a usability-testing laboratory.

Acknowledgements

Thank you to Kyle Goehner who worked with me on the initial project that this research got its inspiration from.

References

[1] Y. Wu and J. Offutt. Modeling and testing web-based applications. *GMU ISE Technical ISE-TR-02-08* 2002.

[2] A. Rukshan and A. Baravalle. Automated usability testing: Analysing asia web sites. *arXiv Preprint arXiv:1212.1849* 2012.

[3] J. Harty. Finding usability bugs with automated tests. *Commun ACM 54(2),* pp. 44-49. 2011.

[4] A. Seffah, M. Donyaee, R. B. Kline and H. K. Padda. Usability measurement and metrics: A consolidated model. *Software Quality Journal 14(2),* pp. 159-178. 2006.

[5] J. Scholtz. Usability evaluation. *National Institute of Standards and Technology* 2004.

[6] W. Hwang and G. Salvendy. Number of people required for usability evaluation: The 10±2 rule. *Commun ACM 53(5),* pp. 130-133. 2010.

[7] A. Sonderegger and J. Sauer. The influence of laboratory set-up in usability tests: Effects on user performance, subjective ratings and physiological measures. *Ergonomics 52(11),* pp. 1350-1361. 2009.

[8] B. Zhou and J. Chen. User behavior based website link structure evaluation and improvement. Presented at ICWI. 2002, Available: http://dblp.uni-trier.de/db/conf/iadis/icwi2002.html#ZhouC02.

[9] D. C. Kung, C. Liu and P. Hsia. An object-oriented web test model for testing web applications. Presented at Quality Software, 2000. Proceedings. First Asia-Pacific Conference On. 2000, .

[10] J. A. JACKO and G. SALVENDY. Hierarchical menu design: Breadth, depth, and task complexity. *Percept. Mot. Skills 82(3c),* pp. 1187-1201. 1996.

[11] D. J. Campbell. Task complexity: A review and analysis. *Academy of Management Review 13(1),* pp. 40-52. 1988.

[12] A. Oztekin, D. Delen, A. Turkyilmaz and S. Zaim. A machine learning-based usability evaluation method for eLearning systems. *Decis. Support Syst.* 2013.

[13] A. Marchetto, R. Tiella, P. Tonella, N. Alshahwan and M. Harman. Crawlability metrics for automated web testing. *International Journal on Software Tools for Technology Transfer 13(2),* pp. 131-149. 2011.

[14] H. Liu, J. Janssen and E. Milios. Using HMM to learn user browsing patterns for focused web crawling. *Data Knowl. Eng. 59(2),* pp. 270-291. 2006.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss and V. Dubourg. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research 12*pp. 2825-2830. 2011.

[16] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. Presented at IJCAI. 1995, .

[17] N. Roy and A. McCallum. Toward optimal active learning through monte carlo estimation of error reduction. *ICML, Williamstown* 2001.