Protecting Blind Screen-Reader Users From Deceptive Content

Anna Dobrenen, bluecat9951@gmail.com
Computer Science at Virginia Wesleyan University

Vikas Ashok, vganjigu@cs.odu.edu
Department of Computer Science, Old Dominion University

Abstract

Visually impaired people who want to use a computer rely on screen readers to independently do this. This research focuses on beginning to build a chrome extension in order to help users more safely navigate the internet using a screen reader. to begin collecting the data, a screen reader was used to help determine items in the website that might take the user somewhere they did not mean to go since the link or image was not sufficiently able to be described by the screen reader. Next, those items were tagged with 'data-attribute="deceptive"'. After, those data-attributes were extracted and tagged with values for various features in it, and a code at the end for if it was a deceptive item. Then six different machine learning models were created in order to predict whether an item on a website is deceptive. Overall, the best model for this data set was the Random Forest Classification from the Scikit-Learn Python Library. Overall, there is much more to be done to improve the accuracy and usability of the models, and then develop the chrome extension, but this is research created a point to begin from for future research.

1 Introduction

Many people know at least one person who has a visual impairment, whether they are an older relative or a friend who relies on glasses or contacts. Despite this fact, many people underestimate the sheer number who have a visual impairment of some variety. According to the World Health Organization, there are approximately 2.2 billion people who have some sort of visual impairment; unfortunately, even with this large number of people, there is truly very little in the way of assistive technology for them[1]. One of the few assistive technologies that exist are screen readers. Screen readers are programs either in a browser, or downloaded to the device that can access and read what is displayed on the screen, then audibly reads that out to the user. With a screen reader, the user is completely dependent upon audio produced by what the program can read from the code and text on the site. Screen reader users are often unable to use visual clues to completely understand what is displayed on the screen due to their disability and screen reader programs often have some level of inaccuracy and don't give complete visual descriptions. The lack of visual clues isn't the only issue that screen reading programs have. Most who are dependent on screen readers cannot rely on the mouse for navigation and so have to use a tactile navigation system like the arrow keys and tab. This method is rather cumbersome to use and doesn't help when many websites have links and additional descriptions that can be difficult to reach when arrow keys and the tab key are the only method of navigation. In addition to the lack of visual context and subpar navigation, screen readers also often read out slower than most people would read. With all of these aspect combined, the user may end up navigating themselves to a malicious website or even downloading a harmful program to their machine. Even in cases without malicious software, many struggle when they accidentally find themselves on another site and are unable to navigate back to the intended URL. To help users with these issues, this research began working on building an intelligent browser extension that will help users navigate the internet and identify deceptive items and keep them safer online. Deceptive items on web pages are parts of the web page such as ads, images, links, and buttons that do not have an accurate description that a screen reader is able to find and read to the user for them to understand what that item is. In order to begin to accomplish this, first a data set must be built. Then, using that information, machine learning classifiers were identified. Using the data set and classifiers, machine learning algorithms were run and then refined. After that the best model is taken and a browser extension will be created from it.

2 Related Work

Related research/ Literature Survey

While there has been research on the accessibility of websites for visually impaired users,there has yet to be research focused on making existing websites accessible without the direct help of the site developers. Examples of the research that has been done on what makes websites accessible can be found in two articles, "Moving toward a universally accessible web: Web accessibility and education"[2] by Serhat Kurt on the state of web accessibility in 2017 and provides recommendations on how to make it more accessible and "The Accessibility of Taif University Blackboard for Visually Impaired Students"[3] by Mrim Alnfiai and Wajdi Alhakami on what difficulties students found while using Blackboard, a website for classes and assignments. In another research project, "Auto-Parsing Network for Image Captioning and Visual Question Answering"[4] headed by Xu Yang, researchers focused on building an algorithm to access hidden HTML trees to aid in creating image captions.

3 Methodology

The first step in this research was to form a data set. To begin building the data set, first an understanding of how screen readers work was needed. For this research, the NonVisual Desktop Access, or NVDA , screen reader was used to navigate through various websites such as news sites, store pages, and blogs in addition

to others. Once what the screen reader could and could not easily identify was determined, deceptive items were tagged with 'data-attribute="deceptive"'. In total, 62 websites were used to build the data set and 286 out of 574 total data points were given the deceptive tag, while the remaining data points were tagged as nondeceptive. After the data points were tagged in the HTML code, the website was saved to be used later in creating the CSV file. Using those tagged data-attributes, features were determined that help identify if something is deceptive or nondeceptive. These features were the presence of alt text, keywords, video tag if the data point is a video graphic, length of alt text, redirect URL to different page/domain, presence of text in images, text length in picture, keywords in image text, size/dimensions of image, presence of the blue ¿ ad symbol, presence of close(x) ad button with no alt text. Then from these features code was created to extract each data-attribute from the HTML code of each website. This was accomplished using the BeautifulSoup Python library for HTML parsing. Once the data-attributes were extracted, a mix of BeautifulSoup attribute parsing and converting the whole code of the data-attribute to a string in order to search within it was used to evaluate and assign a value to the features in each attribute. Then, a code value was added to indicate whether it is a deceptive attribute with a one, and a nondeceptive attribute with a zero based on the data-attribute tag that was in the HTML code. Out of the features listed previously, only presence of alt text, keywords, video tags (if the data point is a video graphic), and the length of alt text features are currently implemented in this code .Then those values were exported to a comma-separated values, or CSV, file to be used later in the machine learning models as the complete data set.

3.1 Model Design

Once the CSV file of the complete data set was created, it was split 80% into a training data set to teach the model the difference between a deceptive data point verses a nondeceptive data point and 20% into a testing set to determine how accurately the various machine learning models could predict whether a data point was deceptive or not. Then, six different machine learning algorithms were run on those data sets using Scikit-Learn Python Machine Learning Library. The different algorithms that were run were Logistic Regression, SVC, K-nearest Neighbor, Decision Tree, Random Forest, and Naive Bayes. Additionally, some hyperparameter tuning was performed on the Random Forest model to determine the best hyperparameters with which the testing set was run with, which was able to raise the scores by a couple percent.

4 Results

The output of each model was evaluated based on the precision, regression, f-1score, accuracy, and AUC. Precision is the percent of the predicted positives out of the total positives, whereas recall is the percent of the actual positive that the model can predict. The f-1 score is based on the precision and recall scores, and is used to balance them if there is a majority of negatives in the data set. The AUC is the area under the ROC curve which plots the relationship between true and false positives. To help visualize the scores of each model, the number of true and false positive, as well as true and false negatives is shown in the confusion matrices in Figure 1. The precision, regression, and f-1 score were calculated using the classification_report method(Figure 2). The accuracy was also calculated in the classification_report, as well as in a separate table with the AUC(Figure 3). For the hyperparameter tuning, only the classification_report was used to evaluate the results(Figure 4).

5 Discussion

Since the overall averages and F-1 scores were fairly high, four out of the six models having scores over 80%, three of those being over 90%, it is time to focus on some of the incorrectly classified data points as seen in the confusion matrices (Figure 1) and potential causes for their incorrect classifications. One of the potential causes for this inaccuracy would be the fact that the list of keywords that the data point is compared to is not an exhaustive list of all the different possible keywords. The keywords that were used mainly revolved around advertisements and sponsorships such as: "ad, sponsored, facebook, twitter, paid." More keywords than those were included in the code, however there are more that could help indicate if

something is deceptive. In the CSV, there were two types of potential incorrectly classified datapoints that are of note. First, there are rows of only zeros in the CSV file of the whole dataset, which most likely cause false negatives in the models. These nondeceptive data points are most likely marked as nondeceptive due to some other feature that has not yet been implemented such as redirect URLs. Inversely, there are data points that have a code value of one, marking them as deceptive, but also another feature column with a value other than zero in the CSV. Potentially these data points are marked as deceptive because of another feature that has not yet been implemented, or maybe because of the way that the keywords are searched by the code; the code tagged something as a key word that was actually a title or even could have been simply part of another word.

6 Conclusion and Future Work

Though this research focused the development of the basis for a chrome extension, there is still more that can be added. In this research, a simple data set was made, and features were determined and then used to teach various machine learning algorithms how to determine if an item on a web page is deceptive or not. There were many issues encountered over the course of this research such as the process of working with BeautifulSoup to extract the data-attributes being difficult at first as one must try to figure out the correct method from BeautifulSoup to use. Additionally, it did not seem as though the program could find all the data points, but later, it was determined that this was because the missing data points were in files other than the main HTML file. However, there is still much more that can be done to improve the accuracy of the machine learning models. This will make the future extension more usable and accurate across a wider range of websites and items in those websites. The next step would be to implement the other features listed in the Methodology section. Additionally, adding more features and expanding the data set would also help to improve the models and future extension. The data set used was made by one person marking potential data points manually, so there is a chance that it is biased in some way. To fix this, an expanded team of researchers would be needed to facilitate the increase of data points put into the data set. Expanding the data set will reduce the chance of arbitrary bias within, which in turn would increase the accuracy of the model and enable to be used on live websites. Once these things have been done and the various machine learning models have been fine tuned, the chrome extension can be built and tested in collaboration with visually impaired individuals in a user study to get feedback and help it to be further improved before releasing to the general public.

References Cited

[1] W. H. Orginization, "Blindness and vision impairment," 2021.

[2] S. Kurt, "Moving toward a universally accessible web: Web accessibility and education," *Assistive Technology*, vol. 31, no. 4, pp. 199–208, 2019. PMID: 29219749.

[3] M. Alnfiai and W. Alhakami, "The accessibility of taif university blackboard for visually impaired students," *International Journal of Computer Science & Network Security*, vol. 21, no. 6, pp. 258–268, 2021.

[4] X. Yang, C. Gao, H. Zhang, and J. Cai, "Auto-parsing network for image captioning and visual question answering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2197–2207, October 2021.

Figures, Tables and Other templates

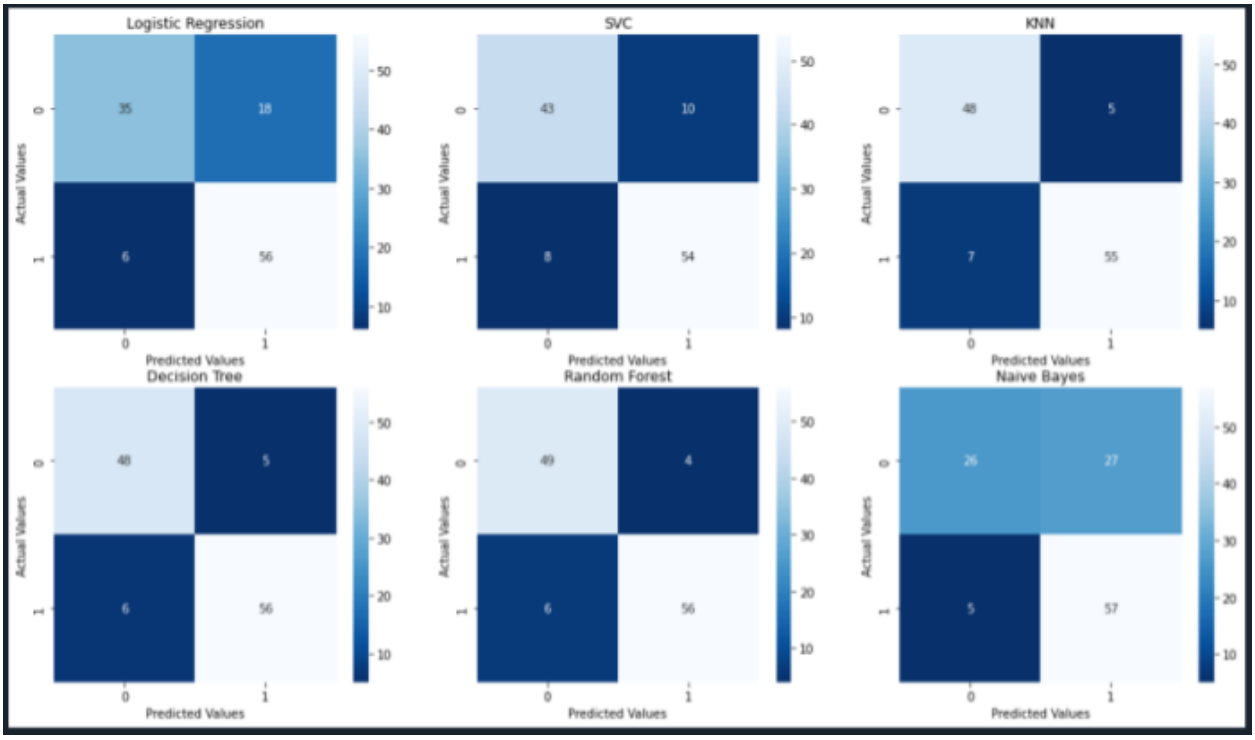

Figure 1: Confusion Matrices showing how each model performed and how many of each kind of prediction there was

Classification Report for Logistic Regression

	precision	recall	f1-score	support
0	0.85	0.66	0.74	53
1	0.76	0.90	0.82	62
accuracy			0.79	115
macro avg	0.81	0.78	0.78	115
weighted avg	0.80	0.79	0.79	115

Classification Report for Decision Tree

	precision	recall	f1-score	support
0	0.89	0.91	0.90	53
1	0.92	0.90	0.91	62
accuracy			0.90	115
macro avg	0.90	0.90	0.90	115
weighted avg	0.90	0.90	0.90	115

Classification Report for SVC

	precision	recall	f1-score	support
0	0.84	0.81	0.83	53
1	0.84	0.87	0.86	62
accuracy			0.84	115
macro avg	0.84	0.84	0.84	115
weighted avg	0.84	0.84	0.84	115

Classification Report for Random Forest

	precision	recall	f1-score	support
0	0.89	0.92	0.91	53
1	0.93	0.90	0.92	62
accuracy			0.91	115
macro avg	0.91	0.91	0.91	115
weighted avg	0.91	0.91	0.91	115

Classification Report for KNN

	precision	recall	f1-score	support
0	0.87	0.91	0.89	53
1	0.92	0.89	0.90	62
accuracy			0.90	115
macro avg	0.89	0.90	0.90	115
weighted avg	0.90	0.90	0.90	115

Classification Report for Naive Bayes

	precision	recall	f1-score	support
0	0.84	0.49	0.62	53
1	0.68	0.92	0.78	62
accuracy			0.72	115
macro avg	0.76	0.70	0.70	115
weighted avg	0.75	0.72	0.71	115

Figure 2: Classification Reports for all models with precision, recall, and F-1 scores

	Model	Accuracy	AUC
0	Logistic Regression	0.791304	0.78
1	SVC	0.843478	0.84
2	KNN	0.895652	0.90
3	Decision Tree	0.904348	0.90
4	Random Forest	0.913043	0.91
5	Naive Bayes	0.721739	0.70

Figure 3: Accuracy and AUC for ML models

0.9217391304347826

{'n_estimators': 1600, 'min_samples_split': 10, 'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth': 80, 'bootstrap': True}

	precision	recall	f1-score	support
0	0.92	0.91	0.91	53
1	0.92	0.94	0.93	62
accuracy			0.92	115
macro avg	0.92	0.92	0.92	115
weighted avg	0.92	0.92	0.92	115

Figure 4: Classification report for the Random Forest algorithm after tuning the hyperparameters