Human-centric Computing and Information Sciences

April 2021 Volume 11

www.hcisjournal.com

KIPS Korea Information Processing Society

KIPS CSWRG Korea Information Processing Society Computer Software Research Group

HCIS | Human-centric Computing and Information Sciences

RESEARCH(Research Manuscript) Open Access

Human-centric Computing and Information Sciences (2021) 11:17
DOI: https://doi.org/10. 22967/HCIS.2021.11.17
Received January 4, 2021; accepted March 23, 2021; published April 15, 2021

Automatic Web Navigation Problem Detection Based on Client-Side Interaction Data

Ainhoa Yera[1,*], Iñigo Perona[1], Olatz Arbelaitz[2], Javier Muguerza[2], J. Eduardo Pérez[2], and Xabier Valencia[3]

Abstract

The current importance of digital competence makes it essential to enable people with disabilities to use digital devices and applications and to automatically adapt site interactions to their needs. Although most of the current adaptable solutions make use of predefined user profiles, automatic detection of user abilities and disabilities is the foundation for building adaptive systems. This work contributes to diminishing the digital divide for people with disabilities by detecting the web navigation problems of users with physical disabilities based on a two-step strategy. The system is based on web user interaction data collected by the RemoTest platform and a complete data mining process applied to the data. First, the device used for interaction is recognized, and then, the problems the user may be having while interacting with the computer are detected. Identification of the device being used and the problems being encountered will allow the most adequate adaptation to be deployed and thus make the navigation more accessible.

Keywords

Accessibility, Adaptive Systems, Web Mining, Machine Learning

1. Introduction

In recent decades, there has been a dramatic increase in the amount of information stored in web servers and in the number of web users. Website access has transformed the way we search for information and the channels we use to communicate and participate in society; therefore, digital competences are currently skills that everyone is presupposed to possess. As a result, it is important to train people with disabilities in using those devices and applications and to adapt site interactions to their needs to diminish the added difficulties they could experience as part of the considerably greater digital divide affecting

***Corresponding Author:** A. Yera (ainhoa.yera @ehu.eus)
[1]Department of Computer Languages and Systems, University of the Basque Country, Bilbo-Bilbao, Spain
[2]Department of Computer Architecture and Technology, University of the Basque Country, Donostia-San Sebastián, Spain
[3]Department of Software and Systems Engineering, University of Mondragon, Arrasate-Mondragón, Spain

people with disabilities.

It is necessary to follow accessibility guidelines when designing interfaces, and most current systems are based on this idea [1]. However, unfortunately, this approach might not be enough to understand human online behavior [2] and to ensure that people with disabilities experience smooth access to a website. In this context, adapting sites in accordance with each user's needs is crucial. One way to determine the type of adaptations a user will benefit from is using specific questionnaires. However, this may not be possible in every situation, and the user may be unwilling to dedicate time to this task. Moreover, user profiles generated by experts will have difficulty identifying all the different types of users who might be interested or might need to navigate in a specific web application [3]. Another option is building adaptive systems that can generate models based solely on in-use information.

The specific adaptations to be applied to the user will depend on her/his characteristics, such as the problems the user could have while navigating. In this context, detection of navigation problems and the type of device being used during in-use time is a compulsory initial stage from which automatic selection and application of the adaptation of the site can be generated and thus is a compulsory stage to improve the user experience.

The advantages of using web mining to attain these objectives are diverse: it is not disruptive; it is based on real navigation data, decreasing the possibility of false assumptions; and it is itself adaptive; i.e., when the characteristics of the user change, the recorded data will also change and will lead to the automatic change of the interaction schema. When adaptations are aimed at a person with physical, sensory or cognitive restrictions, data mining is frequently almost the only feasible way to model user habits or characteristics. However, the difficulty of obtaining data and modeling the interactions makes it difficult to find studies that address this problem.

In that context, this work contributes a system with a two-step architecture to detect user navigation problems while the users are interacting with a website. First, the device being used to interact with the computer while the user is navigating on the web is detected automatically. In the second step, the problem that the user is having in navigating on the site is detected. This system was built based on data collected by RemoTest [4], a tool that collects many and varied data related to user interaction. This process includes a complete data mining process that was designed in collaboration with accessibility experts.

The results showed that the application of a data mining process to interaction data is a promising strategy for automatically detecting user problems and affords an opportunity to provide specific adaptations in the future.

The paper presents some related work in Section 2 and then proceeds in Section 3 with the overall description of the designed system. The platform used to collect the data is described in Section 4, Section 5 describes the database generation process, and in Sections 6 and 7, the device and the problem detection systems are detailed together with their outcomes. Section 8 discusses how some adaptations could be proposed according to the detected devices and patterns. Finally, we draw conclusions and mention the limitations of our work in addition to directions for future research in Section 9.

2. Related Work

When machine learning techniques are applied to user interaction data, the features extracted from the interaction are critical; they will condition the capacity of the machine learning algorithms to solve the problem. MacKenzie et al. [5] defined seven accuracy measures to evaluate human performance with computer pointing devices in point-and-click tasks (target reentry, task axis crossing, movement direction change, orthogonal direction change, movement variability, movement error, and movement offset), which were validated in an evaluation with 12 participants without motor impairments using four different pointing devices: mouse, trackball, joystick and touchpad. Keates et al. [6] introduced six new cursor measures to characterize users with motor impairments on point-and-click tasks, including the distance traveled relative to cursor displacement, the distribution of distance traveled for a range of cursor

speeds, the list of submovements as the number of cycles of cursor acceleration and deceleration, and the cursor distance traveled from the target. Additional measures have also been used to study the performance of people with motor impairments in target selection, such as jerk appearance along cursor trajectory [7] and the distance between click-down and click-up on touch-screen tablets [8].

Almanji et al. [9] presented a review of features extracted from the client-side interaction data of users with pointing devices who suffered from upper-limb impairments due to cerebral palsy.

They proposed a model that measured the influence of the Manual Ability Classification System (MACS) level of each user and the characteristics of the analyzed features. Among the analyzed features, movement time, acceleration-deceleration cycles and average speed were the most significant. The authors claimed that for individuals with cerebral palsy, it is more important to focus on methods to increase speed because they already appear to have sufficient accuracy.

Hurst et al. [10] proposed systems to automatically detect pointing performance with the aim of learning how to deploy adaptations at the correct time without prior knowledge of the participants' ability. They used client-side interaction data to build several systems to (i) distinguish the pointing behaviors of individuals without problems from those of individuals with motor impairments, (ii) distinguish the pointing behaviors of young people from those of people with Parkinson disease or older adults, and (iii) determine the need for the steady-click adaptation that was designed to minimize pointer slips during clicking. All the systems were built on labeled databases, including features related to clicking, features related to movement, pause features and task-specific features, and used wrapper methods to select the features. The C4.5 classifier achieved high accuracy values: (i) 92.7%, (ii) 91.6%, and (iii) 94.4%.

De Santana et al. [11] proposed a remote evaluation tool (WELFIT) for identifying web usage patterns making use of client-side interaction data (event streams) to differentiate event streams generated by people with and without disabilities. They labeled the groups built in a clustering procedure as AT (users using assistive technologies) or non-AT, according to the majority in each group. Aiming to identify web usage patterns within the constructed groups, they found significant differences in the distribution of several features between AT and non-AT users.

Augstein et al. [12] presented a personalized interaction approach where a set of metrics was computed based on different interaction tests performed by 22 users (four of whom had cognitive impairments) and then used to recommend the so-called interaction device setting (IDS) that best fitted the user needs. In particular, the interaction tests were performed using three different IDSs: physical pressure based on smartphone vibration absorption, physical pressure based on smartphone magnetic field manipulation and hand or arm shaking using a smart watch or an armband with an integrated position/acceleration sensor. According to their results, in more than 95% of the cases, the recommended IDS was the one the user had expected.

Perez et al. [13] studied the benefits of two cursor adaptations for assisting web navigation in a longitudinal experiment with eight participants with motor impairments over several weeks. Based on the interaction data recorded from the participants, seven metrics for cursor path evaluation were calculated to study their performance. The features calculated were movement time, pointing time, clicking time, distance traveled, curvature index, number of pauses and target reentry. The results showed significant improvements over the original for both tested virtual cursors in six of the seven features studied.

Martin-Hammond et al. [14] conducted an experiment with 27 participants with pointing problems to inform the design of an adaptive interface for web navigation. By tracking user pointing performance over time, the system provided dynamic notifications and assistance tailored to the user so she/he could choose the preferred adaptations to assist in web navigation.

To our knowledge, no work has analyzed a set of users (with and without physical impairments) interacting with their preferred device (keyboard, trackball, joystick or mouse) and tried to determine any problematic pattern that could happen in any of the two types of tasks defined: mechanical tasks or mechanical and cognitive tasks requiring some cognitive effort.

3. System Description

The system we propose comprises a complete data mining process, summarized in Fig. 1, which consists of four main stages: data collection from designed experiments, data filtering and building the databases to be used in the next two stages, automatic device detection and automatic problem detection.

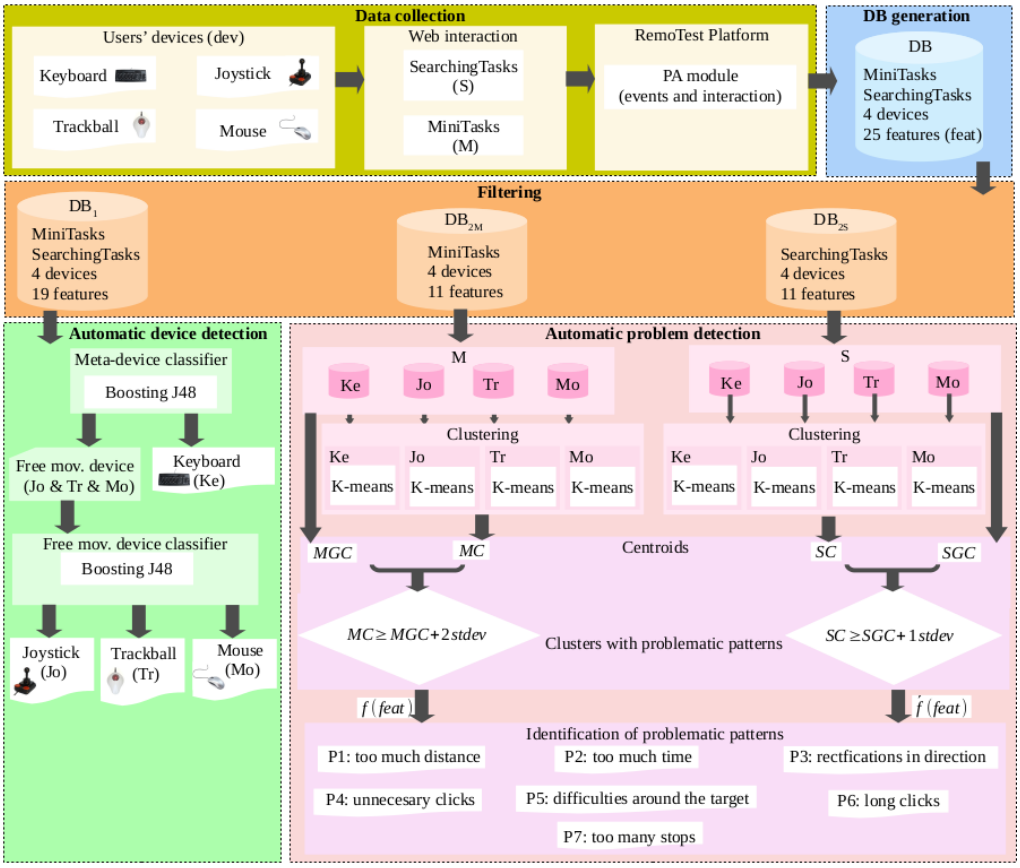


Fig. 1. Description of the designed system.

The data collection stage requires experiments to be designed and users to be recruited. In our case, 15 users carried out two types of tasks, each using her/his preferred device (keyboard, joystick, trackball and mouse), and all the interaction data were recorded using the RemoTest platform. A feature extraction process was carried out to convert the recorded data into the database used for the experiments.

The data and procedures used for device detection and problem detection were different, and consequently, different databases (DB_1, DB_{2M}, and DB_{2S}) were filtered from the original database. Then, two independent subsystems were designed: a two-stage supervised learning system to automatically detect the device used while the user was interacting with the web platform and a subsystem for each type of device and task to automatically detect problems during interaction.

4. Data Collection

RemoTest [4] is a platform with a hybrid client-server architecture that provides tools to define many and varied web-based user experiments. The platform consists of four modules: the Experimenter Module (EXm), Participant Module (PAm), Coordinator Module (COm), and Results Viewer Module (RVm).

Although the RVm is capable of processing and visualizing some of the data, to build the system proposed in this work, we used the raw data recorded by the PAm module, working directly with events, such as cursor movements, key presses, scrolls, and clicks.

Fifteen subjects divided into four groups depending on the input device used for pointing and clicking actions took part in the experiment: four keyboard users (Ke), one trackball user (Tr), four joystick users (Jo), and six mouse users (Mo). Most of the subjects had over seven years of experience of daily computer use, and each of them interacted with the computer using their preferred device or the one that best suited her/his needs. The subjects in the first three groups (Ke, Tr, and Jo) had motor impairments, whereas those in the last group (Mo) had no disabilities.

The experiments were carried out in the laboratory in a controlled environment, and all the participants used the same experimental station with the following characteristics: Dell Precision M6700 laptop, Windows 7 (64 bit) operating system and an additional LCD monitor (aspect ratio 16:10) with a diagonal size of 24 inches and display resolution set to 1,920×1,200 pixels to present stimuli to the participants.

Before starting the experiment, the participants took time to adjust the pointer motion behavior according to their preferences. Subjects from the Ke, Tr, and Jo groups used their own personal input devices, and all the participants in group Mo used the same optical USB mouse provided by the researchers (Dell M-UVDEL1).

4.1 Tasks' Characteristics

Two different websites were selected as stimuli for the experiment: the Discapnet website (https://web.archive.org/web/20161128112949/http://www.discapnet.es:80/Castellano/Paginas/default.aspx), which provides information to people with disabilities, and the institutional website of the Council of Gipuzkoa (http://www.gipuzkoa.eus/). A third informational website about the Bidasoa (http://www.gipuzkoa.eus/) local area was used for training purposes. All three websites claim, in their accessibility sections, to conform to a certain level of the Web Content Accessibility Guidelines (WCAG) (Discapnet to Level AA and Gipuzkoa and Bidasoa to Level A).

The users carried out two types of tasks: MiniTasks (*M*) and SearchingTasks (*S*). Each MiniTask consisted of clicking on one highlighted target. After each target click, the position of the cursor was reset to the center of the screen. These types of tasks have no cognitive phase; they are designed to be straightforward and short navigations to exclusively measure the users' motor skills or intentional navigation. The SearchingTasks consisted of searching for a precise web page on a website after having been given its title. The target web pages were between two and four steps away on the website. These tasks require a cognitive phase, and the intentional navigation starts when the user identifies the target.

In total, each user carried out 156 tasks (144 MiniTasks and 12 SearchingTasks on two websites).

5. Database Generation

After the interaction data were recorded with the RemoTest platform, these data were converted to a labeled database, i.e., to a format suitable for use in the data mining process. In collaboration with the accessibility experts, we selected the features believed to be meaningful for our objectives and extracted them from the collected data. Among the 25 features, some were used in the literature, such as jerk-based features, and others were created with a view to describing the case to be classified.

In the generated database, each instance represents the interaction of a user within a web page in a summarized form. Theoretically, in MiniTasks, there were 2,160 entries or instances (15 users × 144 MiniTasks), and in SearchingTasks, there were approximately 540 examples (15 users × 12 SearchingTasks × 2–4 clicks). To compute the selected features, a minimum activity needs to be ensured within each URL navigation. As a result, only the interactions that met the four following conditions were added to the database: (1) number of MouseMove events $\geq$5 (cursor-move events are recorded

more or less every 10 ms and offer information about cursor position); (2) distance traveled by the cursor ≥0; (3) dimensions of the clicked target area were recorded by RemoTest = true; and (4) number of click-up/click-down events ≥1 (this condition ensures that at least one click has been made).

These conditions diminished the number of recorded entries to 1,713 for MiniTasks (DB_{2M}) and 435 for SearchingTasks (DB_{2S}). The examples were divided into eight databases resulting from the four devices and the two types of tasks, containing a total of 2,148 examples (DB_1), of which 1,713 belonged to MiniTasks (DB_{2M}) and 435 to SearchingTasks (DB_{2S}).

To build the automatic problem detection system, we used and extracted 25 features (Table 1), which were decided in conjunction with the accessibility experts. However, based on the experts' suggestions, the sets of features used for device detection and problem detection were different: 19 features were used for device detection (DB_1), and 11 features were used for problem detection (DB_{2M} and DB_{2S}).

To compute each feature, all the events triggered by the user while visiting the page were taken into account. The features were computed using interaction events of different natures: (i) space events derived from the pointer movements on the page; (ii) action events such as click, key press, scroll or use of the wheel; and (iii) time events considering the time taken to perform the space movements or actions. All the events recorded by the RemoTest platform were indicated in the work of Valencia et al. [4]. They were stored in a MongoDB database, from which feature computation was performed mostly in Java, with R used for signal smoothing functions.

In the process of recording the interaction data, some errors or fluctuations may have appeared that led to outlier navigational behaviors or impossible behaviors. To reduce the effects of these behaviors and obtain a smoother signal, a Butterworth filter configured as a low-pass filter was applied to the following features: cursor speeds (MedSpeed), cursor accelerations (MedAcc), cursor jerks (MedJerk) and angles of cursor direction (NnotHVDMov, NHVMov, NQuadCh, NStrQuadCh, RHVDMov, and RStrQuadCh). In the case of the nominal variables, smoothing was carried out based on a simple moving mode method and using a subset of 11 elements (the five previous elements, the current element and the five subsequent elements).

The diversity of interaction ability among users led us to define a different gap for each user. To be considered a gap, a stop needed to be greater than the median of all stops of the concerned user. This heuristic was used because very short gaps happen through RemoTest capturing events at a frequency of 10 ms, and very long gaps are due to the thinking process of the user and not caused by attempts to rectify the intended navigation.

The analysis of the average values (global centroids) obtained for the features used for problem detection, which was our final goal, split by device and type of task, suggested that the overall behavior is affected by both the type of device used and the type of task. For instance, according to the MiniTasks global centroid, the highest values of MedJerk are observed for the mouse; in contrast, the smallest values correspond to the keyboard. The trackball is prone to register more NQuadCh. In regard to TotTime, keyboard users spend the most time completing the task, while mouse users spend the least time. Regarding NGap, keyboard users make the most stops in navigation. Regarding NCross, mouse and keyboard users are the most skilled in controlling the cursor around the target. The ClickTime is much higher for joystick users.

The trends of the features across devices in the SearchingTask part are similar to those described in the MiniTask part, but the values are higher. Some of the highlighted differences appear in the NGap, ClickTime and ClickDist features. The former becomes equal for all devices when the search process is added, with the trackball, joystick and keyboard devices having more stops.

High ClickTime and ClickDist values denote that the users need more time and distance to make the click; keyboard users need more time (ClickTime), whereas mouse users travel more distance (ClickDist). In addition, increases in the value of RCurDistOpt, the ratio between the traveled distance and the optimal distance, or TotTime denote that SearchingTasks are, as expected, longer and more difficult. In conclusion, the observable differences between devices and tasks suggest that the identification of the device is important for later identifying problems. Consequently, we proposed a two-phase system and

divided the database by device for the second phase.

All the features were standardized (standard scores were calculated) for each device and task to eliminate the bias that their range difference could introduce to the designed classifiers.

Table 1. Description of the 25 features extracted for each page visited

Id	Feature	Unit	Dv	Pr	Description
1	ClickDist	px		x	Average distance between click-down and click-up actions.
2	ClickTime	ms		x	Average time between click-down and click-up actions.
3	CurDist	px	x		Total distance travelled with the cursor.
4	DiagCurArea	px		x	Length of the diagonal of the rectangle that circumscribes the area traversed by the cursor.
5	MedAcc	px/s^2	x		Median of the cursor accelerations.
6	MedGapTime	ms	x		Median of the time intervals without cursor movements (gaps).
7	MedJerk	px/s^3		x	Median of the changes of accelerations of the cursor movements.
8	MedSpeed	px/s	x		Median of the cursor movements speeds.
9	NnotHVDMov	#	x		Number of movements not aligned with horizontal, vertical or diagonal axes.
10	NClick	#	x	x	Number of clicks.
11	NCross	#		x	Number of times the cursor crosses the clickable area limits.
12	NHVMov	#	x		Number of movements aligned with the horizontal or vertical axes.
13	NDMov	#	x		Number of movements aligned with diagonals axes.
14	NQuadCh	#	x	x	Number of quadrant changes in the direction of movements.
15	NEvent	#	x		Number of events recorded by Remotest (cursor-move, click, hover, scroll...)
16	NGap	#		x	Number of gaps.
17	NKeyPress	#	x		Number of keys pressed.
18	NScroll	#	x		Number of times the scroll has been used.
19	NSpKeysPress	#	x		Number of special keys pressed (no letter/digit).
20	NStrQuadCh	#	x		Number of strong changes (two or more quadrants) in the direction of movements.
21	NWheel	#	x		Number of times the wheel has been used.
22	RHVDmov	ratio	x		(NCross+NDMov)/(NCross+NDMov+NnotHVDMov).
23	RStrQuadCh	ratio	x	x	NStrQuadCh/NQuadCh
24	RCurDistOpt	ratio	x	x	Ratio between the CurDist feature and the distance between the initial position of the cursor and the target (optimal distance).
25	TotTime	s	x	x	Total time spent in the page.

Px=pixels for distance, s=second, ms=millisecond, #=number of appearances for counting, Dv=the features used for device detection, Pr=the features used for problem detection.

6. First Step: Automatic Device Detection

The device used by each user in the experiments was known; consequently, we used supervised learning algorithms to detect the device used for navigation. The entire database was given to the machine learning algorithms (DB_1) because, from the device point of view, the task type should be irrelevant. Concerning future adaptations, some accessibility experts suggested that the solutions for the users should differ according to two main groups of devices: those with restricted movements and those with free movements. Accordingly, in this work, we propose a two-level hierarchical approach to discriminate

between devices. In the first level, two metaclasses were defined for classification, placing together the devices with similar behavior: discrete input devices or devices with restricted movements, RestrictedMov (keyboard), and devices with nonrestricted movements or FreeMov (joystick, trackball and mouse). In the second level, devices grouped in the FreeMov metaclass were modeled and classified, i.e., the joystick, trackball and mouse classes.

Classifiers were built and evaluated in Weka [15], where four popular and well-known algorithms were used: naive Bayes (NB), IBK, SVM, and J48, all of which have default parameters. Moreover, two metaclassifiers were used with decision trees (J48) as weak classifiers: bagging (Bagg.) and boosting (Boos.); in both of them, 25 iterations were carried out to create a strong classifier. For validation purposes, a fivefold cross-validation (5-fold CV) strategy was used, with 80% of the database dedicated to training and 20% to testing.

As suggested by the accessibility experts, in the first level of the approach, the classes that had to be perfectly discriminated were addressed to minimize the critical error, whereas in the second level, where the interactions are more similar, the classes with smaller differences were addressed.

The first row (critical) in Table 2 shows that the classifiers are able to discriminate the two metaclasses with a high accuracy. Although all but the NB classifiers obtain high accuracy, the best results (shaded) are obtained with metaclassifiers, especially with boosting J48, with accuracy values of 99.26%. As a result, the critical error is very small.

In the second level, we aimed to discriminate between the devices that convey analog movements: joystick, trackball, and mouse. The second row (free m.) in Table 2 shows that the best value was obtained by again boosting J48, with 90.13% accuracy. Therefore, the two levels of the hierarchy were built using the boosting J48 classifier.

The global error of the automatic device classifier system of two levels is 8.43%. However, the critical error (0.74%) is very low compared to any option implemented facing the classification of the four devices at the same time. We consider this hierarchical solution to be the best one for adapting to the interaction of future users.

Table 2. Accuracy values obtained when discriminating the two metaclasses (critical error)

Level	Accuracy (%) of the classifiers					
	NB	**IBK**	**SVM**	**J48**	**Bagg.**	**Boos.**
First (critical)	90.33	96.06	96.30	98.57	99.04	99.26
Second (free m.)	73.33	80.10	80.10	84.24	87.17	90.13

7. Second Step: Problem Detection

The models described in Section 6 were predictive models built using supervised learning algorithms. However, as described in this section, descriptive models were then created to search for and define interaction problems arising while the users were navigating. Unsupervised learning algorithms were required to achieve this aim.

Clustering algorithms can be used to discover behavioral patterns within the data when no prior knowledge about their structure or class exists. Based on the premise of devices having different values for features ($feat$), the idea was to first perform the clustering for each device (*dev*) and type of task (*S*, *M*) and then automatically select the clusters (*i*, *j*) showing anomalous behavior: those with higher deviation in the 11 features selected by the experts for this task. With this aim, the average behavior of the examples grouped within a cluster, the cluster centroid ($MC_{i,feat}^{dev}$, $SC_{j,feat}^{dev}$), is compared to the overall behavior, or global centroids (MGC_{feat}^{dev}, SGC_{feat}^{dev}).

One of the main parameters of the clustering algorithm is the number of clusters generated to obtain

the best partition. We solved this problem by executing the k-means algorithm [15] with different k values and then making a decision for the best k based on eight cluster validity indexes (CVIs) selected from [16]: Silhouette, Davies-Bouldin variation, Calinski-Harabasz, Davies-Bouldin, COP and the generalized Dunn indexes GD33, GD43 and GD53.

To compare the average behavior of a cluster (*i*) to the overall behavior of all the navigations in the corresponding device (*dev*), all the values of the features (feat) were previously normalized (normal distribution) by device. As shown in Equation (1), cluster centroids ($MC_{i,feat}^{dev}$) deviating by more than or equal to two standard deviations ($2stdev_{feat}^{dev}$) of the global centroid (MGC_{feat}^{dev}) were considered good candidates for identifying navigation problems.

$$MC_{i,feat}^{dev} \geq MGC_{feat}^{dev} + 2stdev_{feat}^{dev} \quad dev \in \{Ke, Jo, Tr, Mo\}, i \in \mathbb{N}, i \leq 15 \qquad (1)$$

Considering the sizes of the databases for MiniTasks (one per device), the different k values we tested for the k-means algorithm were 10, 15, 20 and 25. The average cluster number proposed by the eight CVIs used to evaluate the best partitions for each device was 14.53. Accordingly, we selected the nearest k (k=15) for the k-means used in the final problem detection process carried out in the four MiniTask databases.

7.1 Problematic Patterns and Indicators

Clusters where the values of the features deviated were automatically selected, and we then inferred problematic patterns and their meanings.

- **Pattern 1 (too much distance)**. The indicator is the feature RCurDistOpt, which is a good predictor of a target selection problem: the cursor has traveled a much longer distance than the expected one. However, the causes of the problem are unknown. To explore the cause of the problem, it is necessary to analyze the other features triggered at the same time as RCurDistOpt.
- **Pattern 2 (too much time)**. The indicator is the feature TotTime, which is a good predictor of task completion difficulty: the task has taken more time than expected. However, to explore the cause of the problem, it is necessary to analyze the other features triggered at the same time as TotTime.
- **Pattern 3 (rectifications in direction)**. The indicators are the features NQuadCh or RStrQuadCh, which are triggered at the same as the feature NGap because the adjustment of direction normally requires short stops.
- **Pattern 4 (unnecessary clicks)**. The indicator is the feature NClick. In a straightforward task, it is an unexpected behavior; additionally, it seems to be an indicator of ungainliness.
- **Pattern 5 (difficulties around the target)**. The indicator is the feature NCross, which specifically indicates lack of control and precision in landing on the target. Whenever the target is passed over, there is a need to adjust the direction, so it is usual to see it in combination with Pattern 3. In this context, the user may miss clicking on the target, so it will also frequently appear in combination with Pattern 4.
- **Pattern 6 (long clicks)**. The indicator is the feature ClickTime. In a straightforward task, it is an unexpected behavior; indeed, it seems to be mainly an indicator of indecisive behavior.
- **Pattern 7 (too many stops)**. The indicator is the feature NGap, which is mostly related to the action of rectification (Pattern 3), but there are cases when the user makes no rectification and continues in the same direction, giving the idea that she/he is retaking control of the cursor.

Table 3 summarizes the problematic patterns inferred from the automatically selected clusters and the deviated features in each pattern.

Although the experts initially marked 11 features as suitable indicators of navigation problems, the features MedJerk, ClickDist, and DiagCurArea were not matched with any of the seven problematic

patterns inferred but may be connected to other types of problems not identified in this research. Consequently, their values are not shown in the table that describes the problematic patterns (Table 3) or in the table that shows examples of clusters with problematic patterns (Table 4).

Table 4 shows one of the clusters identified as problematic for each device.

Table 3. Description of the seven problematic patterns inferred and the related features for MiniTasks

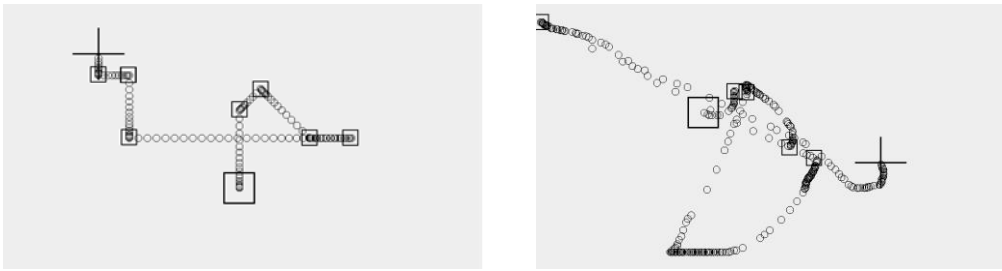
	Pattern 1	Pattern 2	Pattern 3	Pattern 4	Pattern 5	Pattern 6	Pattern 7
RCurDistOpt	x						
RStrQuadCh			[x]				
NClick				x			
NQuadCh			[x]				
TotTime (s)		x					
NGap			x				x
NCross					x		
ClickTime (ms)						x	

x=a significant deviation of a feature, [x]=a deviation of at least one of the marked features (OR condition).

Table 4. Standard deviations of the cluster centroids with problematic patterns (MiniTasks)

Device	Pattern	Centroid: $MC_{i,feat}^{dev}$							
		RCurDistOpt	RStrQuadCh	NClick	NQuadCh	TotTime (s)	NGap	NCross	ClickTime (ms)
Keyboard	1, 2, 3, 7	**2.21**	1.48	0.87	**2.06**	**2.21**	**2.24**	1.78	-0.03
Joystick	1, 2, 4, 5	**4.48**	0.94	**4.52**	**2.35**	**6.77**	**5.61**	**2.58**	0.58
Trackball	6	-0.13	0.41	-0.18	0.15	0.56	1.73	1.00	**2.49**
Mouse	1, 2, 3, 4, 7	**6.51**	1.03	**6.33**	**5.82**	**7.24**	**7.94**	0.77	1.08

The features that deviate more than two standard deviations, marked in bold.



(a) (b)

Fig. 2. Examples of problematic navigation (MiniTasks): (a) keyboard and (b) mouse.

In Fig. 2, we also include as examples some of the navigations of the clusters. Each of the represented navigations is linked to several of the identified patterns, meaning that users having problems are probably experiencing difficulties in many aspects.

In Fig. 2, we present an example of keyboard and mouse where some of the patterns detected in the cluster are visible. The large square represents the starting point, the medium square indicates a cursor stop, the small circumference indicates the cursor's position every 10 ms, the medium circumference shows where a scroll occurred, the medium cross means that a click was made and the large cross indicates the target position. The example for the keyboard corresponds to cluster 7 with evident rectifications in the direction (P3) and excessive stops (P7). In the mouse case, the example corresponds

to cluster 7, where too much distance (P1) is clearly manifested.

7.2 Pattern Discovery in SearchingTasks

Compared to MiniTasks (DB_{2M}), SearchingTasks (DB_{2S}) are not as straightforward; they have a thinking period during which the user has no fixed direction. With regard to the variables, the greatest difference between the two types of tasks is the average time needed to complete them. For the second group (SearchingTasks), the cognitive component takes effect, and consequently, the times increase considerably for all devices, but the rank and differences are maintained for both types of tasks. The values for the rest of the features also vary considerably according to the device. Consequently, we repeated the same process carried out with MiniTasks in the four databases generated for SearchingTasks.

As shown in Equation (2), in this case, cluster centroids ($SC^{dev}_{j,feat}$) deviating more than or equal to one standard deviation ($stdev^{dev}_{feat}$) of the global centroid (SGC^{dev}_{feat}) were considered good candidates for identifying navigation problems. The higher averages and standard deviations of the values of the features in SearchingTasks justified this new threshold (1 stdev).

$$SC^{dev}_{j,feat} \geq SGC^{dev}_{feat} + 1stdev^{dev}_{feat} \quad dev \in \{Ke, Jo, Tr, Mo\}, j \in \mathbb{N}, j \leq 4 \tag{2}$$

Considering the sizes of the databases for SearchingTasks (one per device), the different k values tested for the k-means algorithm were set to $k \leq 4$. We computed and averaged the eight CVIs to evaluate the best partitions for each device, obtaining a value of 3.72 for k. Accordingly, k=4 was used in the k-means. Although the selection was based on the CVIs, the outcome was a set of partitions with similar cluster sizes in both cases. For the MiniTask navigations (partitioned with k=15), on average, 25.9, 32.7, 6.5, and 49.1 navigations per cluster were obtained (keyboard, joystick, trackball, and mouse, respectively), whereas for the SearchingTask navigations, similar cluster sizes were obtained using k=4: 19.0, 31.8, 8.0, and 50.5 (keyboard, joystick, trackball, and mouse, respectively).

The patterns arising in the deviated clusters were mostly identical to those discovered for MiniTasks (see Section 7.1). The only difference appeared for the feature NCross, which never deviated for SearchingTasks; thus, Pattern 5 (difficulties around the target) was not inferred for this context. This is comprehensible because in the MiniTasks, the users must reach particular targets, whereas in the SearchingTasks, they freely decide the targets they want to reach and thus may be more precise in this exercise.

Table 5 shows one of the clusters identified as problematic for each device.

Table 5. Standard deviations of the cluster centroids with problematic patterns (SearchingTasks)

Device	Pattern	Centroid: $SC^{dev}_{j,feat}$							
		RCurDistOpt	**RStrQuadCh**	**NClick**	**NQuadCh**	**TotTime (s)**	**NGap**	**NCross**	**ClickTime (ms)**
Keyboard	2, 3, 4, 7	0.20	0.30	**1.07**	**1.27**	**1.47**	**1.55**	0.61	0.51
Joystick	2, 3, 7	0.75	-0.18	0.19	**1.32**	**1.33**	**1.17**	0.38	0.17
Trackball	4, 6	-0.23	-0.13	**1.56**	0.21	0.49	0.91	0.22	**1.49**
Mouse	2, 3, 4, 7	0.83	-0.04	**1.18**	**2.11**	**1.39**	**1.96**	0.44	0.19

The features that deviate more than one standard deviation from the average, marked in bold.

(a) (b)

Fig. 3. Examples of problematic navigation (SearchingTasks): (a) joystick and (b) trackball.

Some of the patterns are also described in Fig. 3, where one of the navigations of the cluster is illustrated for joystick (left) and trackball (right). The example for the joystick corresponds to cluster 1, where excessive time (P2) and rectifications in the direction (P3) are very perceptible. In the trackball case, the example corresponds to cluster 2, which clearly reveals unnecessary clicks (P4).

8. Use of the Detected Patterns to Facilitate Navigation

The designed system can be used to detect navigation problems in real time. When a user interacts with a web platform, 25 features can be extracted from the data recorded with the RemoTest platform, with 19 used to automatically detect the device used based on the model described in Section 6.1. Then, according to the type of device and task, the nearest cluster (k-NN) of the adequate partition can be used to determine whether the user is having problems and which problems she/he is experiencing.

The set of problematic patterns detected not only describes the accessibility barriers experienced by the users but also enables the definition of the most suitable adaptation techniques to avoid them. With this aim, in this section, potential sources of problematic patterns are analyzed, and feasible solutions are proposed (see the summary in Table 6).

Pattern 1 (too much distance) and Pattern 2 (too much time) occur as consequences of other patterns found in the study. For example, these patterns have a direct connection to Pattern 5 (difficulties around the target), implying that the user requires increased distance and time to select the desired target. Similarly, Pattern 3 (rectifications in direction) can be a consequence of either the handling of the input device used or the specific characteristics of the user. For example, discrete input devices such as keyboards allow the user to move the cursor only in predefined directions: horizontal H_i (where *i* can be E or W), vertical V_i (where *i* can be N or S) and diagonal D_i (where *i* can be NE, SE, SW or NW). When targets cannot be reached directly by following one of these predefined angles, the user must rectify the trajectory (select different angles) to reach the target.

People with uncontrolled movements, e.g., people with cerebral palsy, can experience difficulties in maintaining the position of the hand while they are moving the cursor, leading to a need for rectifications in the cursor path (Pattern 3) and producing more stops during the target selection (Pattern 7). In addition, when the target size is not adequate, uncontrolled movements can make target selection difficult (Pattern 5). A lack of control can also provoke involuntary movements during the clicking process, moving the cursor away from the target and therefore generating unnecessary clicks (Pattern 4). In addition, these users might have difficulty pressing the buttons of the joystick, trackball or mouse to perform a click action if they have difficulty stopping the ongoing action, producing long clicks (Pattern 6). This issue can cause clicks outside the target (Pattern 4) or movement of the cursor during the click, preventing the click event from being triggered.

Different strategies can be followed to allow the user to make effective target selections. For instance, the *bubble cursor* technique increases the cursor selection area, depending on the number of selectable targets within reach [17]. An alternative solution is a *magnetic target* that attracts the cursor [18]. In this case, when the cursor is near a target, the cursor is pulled towards the target center, making its activation

easier. Another possible adaptation is so-called *goal crossing* [19], which activates the target when the cursor crosses it. This adaptation makes the clicking action unnecessary, and therefore, the selection of small targets becomes easier. The *cross cursor* technique divides the screen into zones by means of a crossbar that allows a remote target to be selected by typing one letter, indicating the coordinates of the zone. This procedure reduces the required number of corrections and stops. The *goal-crossing* technique appears to be appropriate for people who use trackballs since it does not require handling precision and minimizes the use of buttons. On the other hand, joystick users can benefit from both the *magnetic* and *bubble cursor* techniques, and keyboard users usually prefer the *cross cursor* technique since it helps to reduce the effort to select the target. Therefore, adequate techniques to help to reduce the occurrence of Patterns 3, 4, 5, and 7 can be selected depending on the device used.

With regard to problems relating to Pattern 6, diverse techniques can be used, for instance, *click on down*, *click on up*, *steady click* or *goal crossing*. When *click on down* is used, the target is selected just when the user presses the button. Similarly, *click on up* selects the target when the user releases the button, whereas *steady click* "freezes" the cursor during the click, enabling target selection even if the user moves the cursor away from the target [20].

Since each pattern has diverse associated techniques to avoid the related accessibility barriers, the selection of the most appropriate one for a specific user depends on her/his particular characteristics. Some can be selected by the users themselves when presetting their interfaces, but they can also be selected by adaptive systems depending on the detected device and problem.

Table 6. Summary of the adaptations suggested for problematic patterns for each device

Adaptation		Device			Problematic pattern				
Name	**Ref.**	**Joystick**	**Trackball**	**Keyboard**	**Pattern 3**	**Pattern 4**	**Pattern 5**	**Pattern 6**	**Pattern 7**
Bubble cursor	[17]	X			X	X	X		X
Magnetic target	[18]	X			X	X	X		X
Goal crossing	[19]		X		X	X	X	X	X
Cross cursor	[13]	X	X	X				X	
Click on down		X	X	X				X	
Click on up		X	X	X				X	
Steady click	[20]	X	X	X				X	

(a) (b) (c) (d) (e) (f) (g)

Fig. 4. Pictures of the adaptations suggested for problematic patterns: (a) bubble cursor, (b) magnetic target, (c) goal crossing, (d) cross cursor, (e) click on down, (f) click on up, (g) steady click.

Table 6 and Fig. 4 summarize and show the adaptations discussed for each problematic pattern, considering the device used. When Patterns 3, 4, 5, and 7 appear (direction rectification, unnecessary clicks, target difficulties and too many stops), particular adaptations are recommended for each device, whereas device-independent solutions are recommended for Pattern 6 (long clicks). Figures representing each of the proposed adaptations are shown at the bottom of Table 6, and their use is marked with the corresponding letter.

9. Conclusion and Further Work

This work contributes to diminishing the digital divide for people with disabilities by analyzing the interaction of a set of users (with and without physical impairments) with their preferred device (keyboard, trackball, joystick or mouse) to determine any problematic pattern that could happen in mechanical tasks or tasks requiring both mechanical and cognitive effort. This study proposes a two-step system that first, automatically detects the device used to interact with the web and second, automatically identifies web navigation problems. This detection system includes a complete data mining process applied to web user interaction data collected by the RemoTest platform.

The device being used was detected through a two-level hierarchical approach. At the first level, two metaclasses were discriminated: devices with restricted movements or discrete input devices (keyboard) and devices with free movements (joystick, trackball, and mouse), and an accuracy value of 99.26% was obtained. At the second level, the system distinguished each of the devices with free movements with an accuracy value of 89.84%. Minimizing the critical error at the first level is vital because future interaction adaptations will be very similar for free movement devices and very different from those proposed for discrete input devices.

To detect the problems that users may experience while interacting with the computer, two types of tasks were considered: MiniTasks (directed navigation) and SearchingTasks (searching + directed navigation). For each task and each device, the k-means clustering algorithm was run, and clusters with high probabilities of containing problematic navigation patterns were then automatically filtered based on particular standard deviation thresholds for a set of 11 meaningful characteristics. Then, a visual analysis of their navigation traces was performed to identify different problematic patterns. A total of seven problematic patterns—too much distance, too much time, rectifications in directions, unnecessary clicks, difficulties around the target, long clicks, and too many stops—were found in MiniTasks, and all of these, except Pattern 5 (difficulties around the target), were also observed in SearchingTasks, which is understandable given that in the first scenario, the users had to reach the targets appearing on the screen.

Finally, the paper discusses the hypothetical reasons behind the detected patterns and suggests the most suitable adaptation technique based on these patterns and the devices used.

Unfortunately, having built the system based on client-side data limits the work in two senses: client-side data are required to be used with new users and trigger the most adequate adaptations according to the detected device and problem. In addition, the amount of data that could be used to build the system was also limited, as the data had to be collected in controlled experiments.

On the other hand, the system is able to detect the device used and problems the users are experiencing, and we suggest adequate adaptations, although the adaptations have not yet been activated.

In the near future, we aim to carry out new experiments where both the device used and the problematic patterns are discovered automatically and the corresponding adaptation techniques are activated accordingly. New navigations should be assessed and compared to those without adaptation.

Competing Interests

The authors declare that they have no competing interests.

Funding

This work was funded by the following units: first, the University of the Basque Country UPV/EHU (No. PIF15/143); second, the research group ADIAN, which is supported by the Department of Education, Universities and Research of the Basque Government (No. IT980-16); and finally, the Ministry of Economy and Competitiveness of the Spanish Government, cofounded by the ERDF (PhysComp Project, No. TIN2017-85409-P).

Author’s Contributions

Conceptualization, OA, JM. Investigation and methodology, AY, IP. Project administration, OA, JM. Resources, JEP, XV. Writing of the original draft, AY, IP. Writing of the review and editing, OA, IP, JEP, XV. All authors read and approved the final manuscript.

Acknowledgements

The authors are grateful to the Elkartu coordinating federation of people with physical disabilities and especially to the participants in this study.

References

[1] A. Aqle, K. Khowaja, and D. Al-Thani, "Preliminary evaluation of interactive search engine interface for visually impaired users," *IEEE Access*, vol. 8, pp. 45061-45070, 2020. https://doi.org/10.1109/ACCESS.2020.2977593

[2] C. Cepeda, R. Tonet, D. N. Osorio, H. P. Silva, E. Battegay, M. Cheetham, and H. Gamboa, "Latent: a flexible data collection tool to research human behavior in the context of web navigation," *IEEE Access*, vol. 7, pp. 77659-77673, 2019. https://doi.org/10.1109/ACCESS.2019.2916996

[3] A. Dillon, "Beyond usability: process, outcome and affect in human computer interactions," *Canadian Journal of Information Science*, vol. 26, no. 4, pp. 57-69, 2001.

[4] X. Valencia, J. E. Perez, U. Munoz, M. Arrue, and J. Abascal, "Assisted interaction data analysis of web-based user studies," in *Human-Computer Interaction – Interact 2015*. Cham, Switzerland: Springer, 2015, pp. 1-19. https://doi.org/10.1007/978-3-319-22701-6_1

[5] I. S. MacKenzie, T. Kauppinen, and M. Silfverberg, "Accuracy measures for evaluating computer pointing devices," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* Seattle, WA, 2001, pp. 9-16.

[6] S. Keates, F. Hwang, P. Langdon, P. J. Clarkson, and P. Robinson, "cursor measures for motion-impaired computer users," in *Proceedings of the 5th International ACM Conference on Assistive Technologies*, Edinburgh, Scotland, 2002, pp. 135-142.

[7] J. O. Wobbrock, K. Shinohara, and A. Jansen, "The effects of task dimensionality, endpoint deviation, throughput calculation, and experiment design on pointing measures and models," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Vancouver, Canada, 2011, pp. 1639-1648.

[8] X. Valencia, J. E. Perez, M. Arrue, J. Abascal, C. Duarte, and L. Moreno, "Adapting the web for people with upper body motor impairments using touch screen tablets," *Interacting with Computers*, vol. 29, no. 6, pp. 794-812, 2017.

[9] A. Almanji, T. C. Davies, and N. Susan Stott, "Using cursor measures to investigate the effects of impairment severity on cursor control for youths with cerebral palsy," *International Journal of Human-Computer Studies*, vol. 72, no. 3, pp. 349-357, 2014. https://doi.org/10.1016/j.ijhcs.2013.12.003

[10] A. Hurst, S. E. Hudson, J. Mankoff, and S. Trewin, "Automatically detecting pointing performance," in *Proceedings of the 13th International Conference on Intelligent User Interfaces*, Gran Canaria, Spain, 2008, pp. 11-19.

[11] V. F. de Santana and M. C. C. Baranauskas, "WELFIT: a remote evaluation tool for identifying web usage patterns through client-side logging," *International Journal of Human-Computer Studies*, vol. 76, pp. 40-49, 2015. https://doi.org/10.1016/j.ijhcs.2014.12.005

[12] M. Augstein, T. Neumayr, W. Kurschl, D. Kern, T. Burger, and J. Altmann, "A personalized interaction approach: motivation and use case," in *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, Bratislava, Slovakia, 2017, pp. 221-226.

[13] J. E. Perez, M. Arrue, X. Valencia, and J. Abascal, "Longitudinal study of two virtual cursors for people with motor impairments: a performance and satisfaction analysis on web Navigation," *IEEE Access*, vol. 8, pp. 110381-110396, 2020. https://doi.org/10.1109/ACCESS.2020.3001766

[14] A. Martin-Hammond, F. Hamidi, T. Bhalerao, C. Ortega, A. Ali, C. Hornback, C. Means, and A. Hurst, "Designing an adaptive web navigation interface for users with variable pointing performance," in

Proceedings of the Internet of Accessible Things, Lyon, France, 2018, pp. 1-10.

[15] P. Bhatia, *Data Mining and Data Warehousing: Principles and Practical Techniques*. New York, NY: Cambridge University Press, 2019.

[16] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Perez., I. Perona. "An extensive comparative study of cluster validity indices," *Pattern Recognition*, vol. 46, no. 1, pp. 243-256, 2013. https://doi.org/10.1016/j.patcog.2012.07.021

[17] T. Grossman and R. Balakrishnan. "The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Portland, OR, 2005, pp. 281-290.

[18] J. Park, S. H. Han, and H. Yang, "Evaluation of cursor capturing functions in a target positioning task," *International Journal of Industrial Ergonomics*, vol. 36, no. 8, pp. 721-30, 2006. https://doi.org/10.1016/j.ergon.2006.05.004

[19] J. O. Wobbrock and K. Z. Gajos. "Goal crossing with mice and trackballs for people with motor impairments: performance, submovements, and design directions," *ACM Transactions on Accessible Computing (TACCESS)*, vol. 1, no. 1, pp. 31-37, 2008. https://doi.org/10.1145/1361203.1361207

[20] S. Trewin, S. Keates, and K. Moffatt, "Developing steady clicks: a method of cursor assistance for people with motor impairments," in *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, Portland, OR, 2006, pp. 26-33.