# Impairment Detection using Support Vector Machines

**Jordan Walker**
Department of Computer Science
University of Wisconsin-Madison
Madison, WI 53715
jwalker@cs.wisc.edu

**Abstract**

I attempt to detect impairment in users by subjecting them to a typing test. If the result of the test shows that the user is classified as impaired, this will be presented to the user to allow them to decide whether to continue. The technique used to classify a user is a support vector machine. I find that my classifier works well on this simple model, though the data is not filled out enough to predict completely its usefulness.

## 1. Introduction

Oftentimes a computer user will attempt to accomplish a task when he would be better suited to wait until they can completely focus on the task. Such cases include trying to multi-task, working when too tired, or doing something while intoxicated. There are many reasons a person would want to prevent himself or others from working without the necessary focus.

The detection of these impairments comes down to classification of a stream of characters. If the user is typing text that differs from his normal typing in a noticeable way, this document will be classified as impaired. The question of what to do if a user is impaired is dependent on the application. Sometimes the best response is to lock them out and tell them to continue when they can do so unimpaired. I also see this being useful as just a warning, or possibly a trigger for a different type of test to determine whether the user is impaired in any way.

In this paper I examine how using support vector machines to classify a user as impaired/unimpaired based on text they enter. I will describe the data collection, the design of the feature vector, and the results of this classifier based on the data collected.

## 2. Previous Work

There are several attempts to identify users by their typing style. I will discuss how this can be useful more in the discussion section. The topic of user identification yields the paper Digital Fingerprints(Rehmeyer 2007) which attempts to look at features in the language of users to enforce some security rules. The system builds up a users regular typing patterns and then when they type a password it verifies that the user typing in the password is the same user. I also got some inspiration from a Gmail labs feature which requires a user to perform a series of math tasks before sending an email during certain times of the week.

The use of support vector machines to accomplish this goal of classification is supported by the others(Joachims 1997) who have used it for similar tasks. Categorizing text input can be used to gain many types of information, in my case it is used to gather the state of the user typing on the keyboard.



## 3. Feature Selection

Designing the features to store the data is a difficult task. To capture all the data I had to think about how an impaired person might express that when typing. The speed at which they type, as well as the errors they make are crucial to understanding the state of the user. As a result the feature vector captures the stream of characters typed and the time taken between characters.

In order to retain the information of the layout of the keyboard, the vector is split up by character. The impairment might affect the user by causing them to make a mistake on a certain character (typing a comma instead of a period for example) and this information should be stored. The fraction of mistakes to correct keystrokes may show areas on the keyboard where the user had a hard time (left side of keyboard for right handed typing maybe). Lastly, backspaces are counted to capture whether or not a user is correcting those mistakes made. The type of impairment will determine which features contain the useful information; I will come back to this in the discussion section.

The last portion of the feature vector is timing information. Between each keystroke there elapses a certain amount of time that is recorded in milliseconds. At the end of the test, the mean and variance of the timings is stored in the vector. The mean describes approximately how long it took to for the user to complete the task, while the variance gives a sense of how consistently the user kept that pace (how much they varied from the mean). A distracted user may introduce high variance in times, and this is captured in the vector.

## 4. Data Collection

Central to this task of impairment detection is a program that will collect the data in question. The program displays a test to the user and captures the character stream from the keyboard. Perl is used to tie all the pieces together, using the SVM module for classification and Curses to display and get characters. The feature vector is calculated after the user finishes, after which it is written to file. This growing file of feature vectors can be trained to detect certain types of impairments if the user carefully trains it.

The support vector machine is trained with all data collected in the dataset and the lib-SVM model file is written. A linear kernel works best in my tests, though there was no loss in accuracy compared to the polynomial kernel. Because each test took a relatively long time to complete, I was only able to collect 38 data points. I would prefer to have more data, as the conclusions drawn from the data are not as strong as they otherwise would be.

Another point that should be brought up is that during data collection it was easy to be unimpaired, but collecting impaired data resulted in a self-imposed constraint rather than true impairment. Typing with one hand instead of two may be realistic (while eating, holding a baby, etc.), but it does not represent well other types of impairment. Several points in the dataset were collected while truly impaired (nearly falling asleep while typing). These are in the minority though, so the one handed typing test represents most of the impaired data. The impact of this will be explored further in the results section, but this form of data collection most likely introduced some form of bias to the data.

## 5. Results

At this point there is a dataset with 38 data points. I ran a leave-one-out cross-validation on this dataset to see how well the support vector machine runs. With a linear kernel to train the SVM, it gives an accuracy of 100%. As expected the radial kernel did not work well on this dataset, resulting in an accuracy only slightly higher than 50%. The impression that this method works perfectly is troubling though. I suspect that a couple features dominate the machine, and if a data point were to rely on other features for correct classification (indistinguishable in the dominant features), it would be missed by this model.

To show that this is the case, I calculated the weights of the features in the support vector. The mean time taken feature has a weight of $9.7 \times 10^{-5}$ and the total time is $5.52 \times 10^{-4}$. All other features are negligible in their weight. This anomaly is more from the training set than the feature design. The weights of the character features would likely be significant if the test varied more in time. The strong reliance on time as a feature comes because it is much slower to type with the one handed impairment I imposed on myself.

I would like to see what would happen if I removed the mean time feature and only left the variance. I did not have the time to run this, but I suspect that it would only be useful as a feature some of the time, while the rest of the time, the errors in typing would dominate the result. I will leave this tweaking of the features to future work.

## 6. Discussion

Which features contain information about which impairments is something I had to think about when looking at the results. Mean time seemed to be the dominant feature in my one-handed vs. two-handed dataset. But if I had instead done more tests while distracted (perhaps watching a TV show, or talking to a roommate), then the time may not have differed so drastically. In this case the errors in the character stream might have dominated. There is also the feature of the count of backspaces. This may not hold much information most of the time, but when a user does not use the backspace key at all, or uses it a lot, that should say something about their impairment level. A study of many users and how they type while impaired would be interesting to look at for this type of information.

Some of the ways that this project could be extended would have some very interesting applications. Continually monitoring a user's typing could be used to detect an imposter or other security breach. Requiring a user to verify they are unimpaired before performing a task would ensure quality control and enforce a level of high productivity. These extensions are somewhat distant from the current state of this project, but getting there would only require some tweaking and tuning of the approach. The feature vector should be adjusted to be more mindful of the application in which it is used as well.

I believe that this type of system could be put in place to allow the computer to learn more about minor changes in user behavior. When a user wants to use this information to keep themselves honest, so to speak, then the power of this research can really be found. The Gmail labs test is one use of this idea that shows the validity. If a user doesn't want to embarrass himself by sending an email when intoxicated, he can put the math test in place to prevent it. I picture with the impairment detection presented here, the user could enable the monitoring so that when typing the email the text gets classified as from an impaired user, a warning would pop up preventing them from sending that email. This is just one area in which this method could be applied successfully.

## References


Joachims, T. 1997. Text categorization with support vector machines: Learning with many relevant features. Technical Report 23, Universität Dortmund, LS VIII-Report.

Rehmeyer, J. 2007. Digital fingerprints. *Science News* 171(2):26–28.