# Medical Device Recall Classification

**Sebastian Thiem**
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ 15213
`sthiem@email.arizona.edu`

## Abstract

This work sought to explore novel methods of classifying hardware, software, and security threats in device recall information. Previous work utilized key-word search based approaches, which worked well for hardware and software classification, but was not as effective when trying to classify security threats. Using a more complex language model like a weighted dictionary showed an increase in classification accuracy across all three classes.

## 1 Introduction

### 1.1 Problem Statement

The FDA maintains a database of medical device recalls, which can be used to observe trends in device malfunctions throughout the years. This research chose to focus on computer-related recalls, hardware and software, as well as security threats in these device malfunctions.

Previous work by Alemzadeh et al. used key-word based searches, to classify safety threats due to hardware and software related device failures (1). By analyzing word frequencies amongst a hand-labeled set of hardware and software recalls, a list of the top key-words for each classification was conceived (see Table 1). This method was shown to be effective at classifying hardware and software, however in replicating their methodology it was found that this approach was not very successful in classifying security threats. Using Alemzadeh's approach to classify security threats resulted in only a 47.6% accuracy, as well as 87.4% accuracy for hardware and 71.8% for software.

This research sought to explore novel classification approaches in order to classify security threats in medical device recall data and improve classification for hardware/software faults.

Table 1: Hardware and Software Key-words, Alemzadeh et el. (1)

| **Fault class** | **Key-words** |
|---|---|
| Software | Software, application, function, code, version, backup, database, program, bug, Java, run, upgrade |
| Hardware | Board, chip, hardware, processor, memory, disk, PCB, electronic, electrical, circuit, leak, short-circuit, capacitor, transistor, resistor |
| Battery | Error, system, fail, verification, self-test, reboot, Web, robotic, calculation, document, performance, workstation |
| I/O | Battery, power, supply, outlet, plug, power-up, discharge, charger |

## 1.2 Related Work

### 1.2.1 Analysis of Safety-Critical Computer Failures in Medical Devices

In this paper Alemzadeh, Iyer, Kalbarczyk, and Raman used the FDA's device recall database to observe trends in safety issues related to hardware and software malfunction is medical devices. In this work, they manually determined the most frequent key-words associated with hardware and software malfunctions, and used those keywords to observe computer related safety threats.

These searches were purely key-word based. If a keyword from their list showed up in a device recall that recall was classified according to Table 1. This simple and efficient methodology worked well for their classification problem, and was what brought rise to this current work in security threat analysis.

### 1.2.2 Classification of Security Threats in Information Systems

In order to classify security threats it was necessary to research how security threats are identified. In this work Jouini and Rabai created a tree to classify security threats. The tree itself went into more detail than would be required for the scope of this research, however the leaves of the tree had a lot to say about security classification:

- Destruction of Information
- Corruption of Information
- Loss of Information (Theft)
- Illegal Usage
- Disclosure of Information
- Denial of Use
- Elevation of Privilege

This list gave a clear set of events to monitor the recall data for, and was used as a reference when labeling recalls as potential security threats.

## 2 Methods

### 2.1 Labeling The Data

Before any machine learning model can be trained there will need to be a significant number of labeled samples. In the case of the FDA's recall data, none of the samples are labeled. In order to even begin to perform some semi-supervised learning approaches, at least a few hundred samples needed to be labeled.

Hand labeling samples would take away time from researching other models, so an idea came to mind. Rather than sink time into labeling hundreds of samples by hand; use a model that gets stronger as more labels are added to the data. Many semi-supervised models have this capability, but the most efficient model for real time updates was a weighted dictionary approach.

With this approach in mind, a GUI environment was constructed that allowed for real time classification and labeling (See figure 1). The system worked as follows:

#### 2.1.1 Assigning a Label

The check boxes label the data, and the two submit buttons would save that label to the dataset and then find another unlabeled sample. The reason for having two buttons was so that the user could specify whether they wanted to receive a sample that the classifier was as close to 100% certain of or a sample that was closer to 50% certain. The uncertain data points are the ones the classifier needs labeled the most, so this helped the classifier grow faster. Displaying a confident sample was a simple heuristic way of showing the user how well the classifier is performing.

### 2.1.2 Discriminating Between Classes

The three labels being added to these samples are software failure, hardware failure, and security threat. There is a drop-down box for changing what class is being searched for, using the confident/uncertain mechanic explained previously. This lets the user decide what labels to focus on.

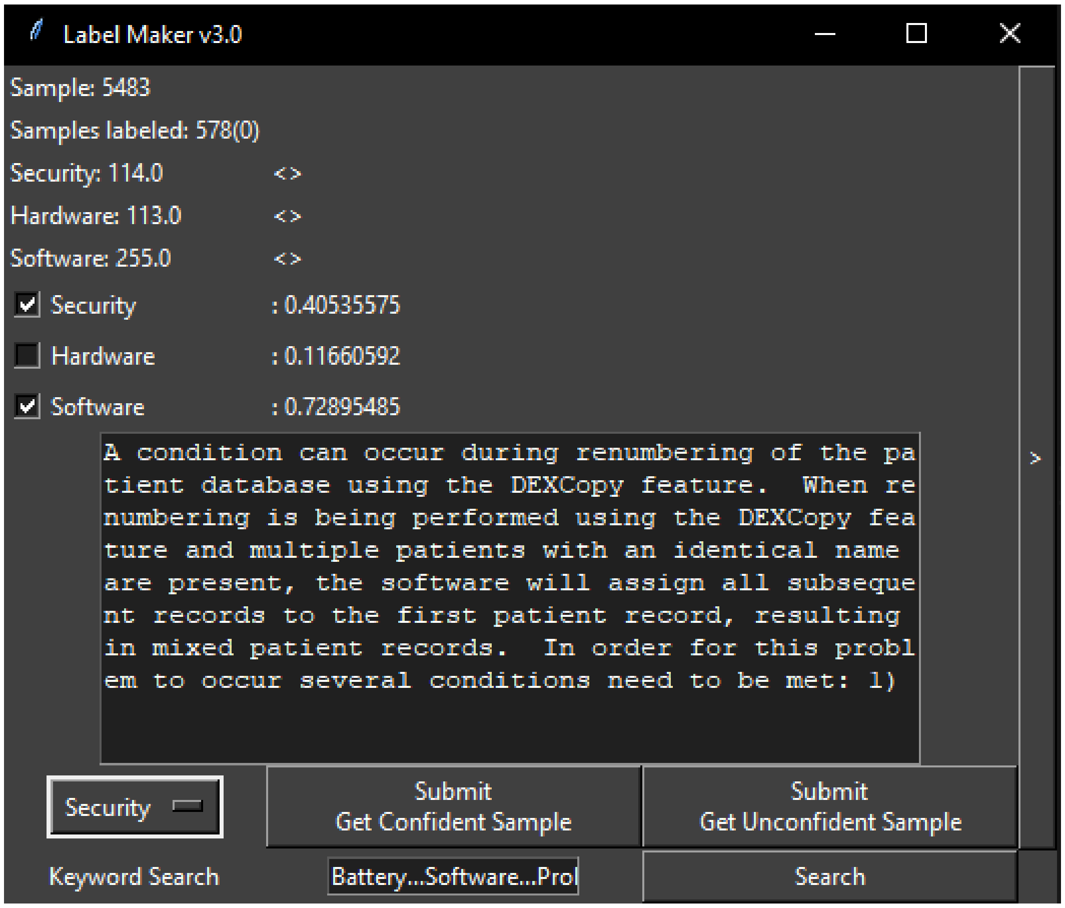
### 2.1.3 Keyword Search

This feature was added to jump-start the model when the dictionary was first growing. Before critical words were introduced into the dictionary the search functions were just guessing. By searching for samples with specific key-words from Alemzadeh's table (1) the user can help the classifier build the dictionary with known computer words.

### 2.1.4 Presenting The Samples

The current sample is displayed in the center textbox. Metrics about the sample and the model are displayed on the top of the interface. Of note here are the frequencies of the samples labeled so far, and the weighted classification of the current sample by the check-boxes. This showed the user how the classifier weighted the sample they are currently seeing. This real time classification was the reason behind the weighted dictionary model chosen.

The empty <> are placeholders for the real time performance analysis. As data points are labeled during runtime, the accuracy of the model's classifications will be measured. This enables the online performance analysis needed for more reliably testing the model.



```
Sample: 5483
Samples labeled: 578(0)
Security: 114.0        <>
Hardware: 113.0        <>
Software: 255.0        <>
[✓] Security           : 0.40535575
[ ] Hardware           : 0.11660592
[✓] Software           : 0.72895485
```

```
A condition can occur during renumbering of the pa
tient database using the DEXCopy feature.  When re
numbering is being performed using the DEXCopy fea
ture and multiple patients with an identical name
are present, the software will assign all subseque
nt records to the first patient record, resulting
in mixed patient records.  In order for this probl
em to occur several conditions need to be met: 1)
```

Keyword Search: Battery...Software...Prol

Figure 1: Label Making Interface
`https://github.com/SeaBass917/label-maker`

### 2.2 Weighted Dictionary

This weighted dictionary builds a dictionary in real time that stores the frequency of a word and the frequency of that word in the three classifications: hardware, software and security. This dictionary can be used to classify a sentence by the following equation:

$$W(S; D) = \frac{1}{|S \cap D|} \sum_{w \in S \cap D} \frac{f_c(w)}{f(w)} \tag{1}$$

Here S is the set of words in the sentence and D is the dictionary. $S \cap D$ are simply the words in the sentence that are also in the dictionary. Each word has 4 values stored alongside it in the dictionary:

- $f_c(w)$ – the frequency of the word in sentences labeled with the given class, one value for each class.
- $f(w)$ – the total frequency of the word among all samples observed.

As the program runs and the user labels the recall data, the frequencies are updated. Words that are not in the dictionary yet are added. After labeling over the 578 samples the dictionary contained 2899 words. All these words and frequencies are stored locally next to the semi-labeled database, so that they carry forward the next time the application starts up.

Some basic pre-processing was done to each sentence before parsing such as the removal of stop words (the, at, it, ect...), removing non-alphabetic characters, and setting all uppercase characters to lowercase.

## 3 Tests & Analysis

In order to analyze the model two tests were set up. The first test monitored the integrity of the model, and the second test was for the performance.

### 3.1 Test 1

The first test monitored semantic drift. As the dictionary grows and more samples are labeled, it is possible for the weights to begin to lose meaning. Too many non-computer related samples could begin to skew the meaning of more generic words that were not filtered out by the pre-processing.

This test was performed by analyzing the performance of the model on the set of labeled samples so far. If this performance is too far from 100% then the model is either not actually improving as labels are added, or the weights of some words are beginning to taint the classification of a sentence. For comparison, predictions were also made using Alemzadeh's key-word based approach.

Table 2: Offline Performance Analysis

| **Class** | **Weighted Dictionary Accuracy** | **Key-Word Accuracy** |
|---|---|---|
| Security | 97.9% | 47.6% |
| Hardware | 98.4% | 87.4% |
| Software | 99.1% | 71.8% |

### 3.2 Test 2

Test 2 sought to analyze the online performance of the model, how it behaved in real time as new labels were being added. This analysis was done by comparing the label added by the user to the prediction the model had for that sample.

For each class: 20 certain and 20 uncertain samples were labeled in 6 isolated rounds of analysis. The results can be seen in Tables 3 and 4 respectively, along with the key-word based accuracies.

Table 3: Online Confident Performance Analysis

| **Class** | **Security** | **Hardware** | **Software** | **Key-Word** |
|---|---|---|---|---|
| Security | 65.0% | 100% | 95.0% | 47.6% |
| Hardware | 100% | 85% | 100% | 87.4% |
| Software | 71.4% | 100% | 95.0% | 71.8% |

Table 4: Online Un-certain Performance Analysis

| **Class** | **Security** | **Hardware** | **Software** | **Key-Word** |
|---|---|---|---|---|
| Security | 60.0% | 100% | 100% | 47.6% |
| Hardware | 95.0% | 55.0% | 95.0% | 87.4% |
| Software | 85.0% | 95.0% | 40.0% | 71.8% |

### 3.3 Analysis

Test 1 did not suggest that the model was suffering from any semantic drift. These accuracies are still considerably high, meaning the model is learning quickly from new samples and is not forgetting how to classify old samples. If these accuracies start to drop as more samples are labeled, this will signal the beginning of a semantic drift in the dictionary.

Test 2 revealed a lot of the weak points in the current model. When isolating the security class searches classifications seem to be closer to a guess than actual classification. There is a very good reason for the model guessing for security threats, and that is lexical overlap.

When the model fails to predict a security threat the predictor's weight is consistently right on the boundary at about 40-47%, however for hardware recalls the weights are near zero. 100% of the security related recalls are all software based. This connection between the software class and the security class creates a dependency between the two. Many of the words used in software recalls will overlap with words in security recalls. 44% of the software recalls were security threats, and as a result whenever a software related recall is presented, the model will weight it as a security threat with around a 44% certainty every time.

The overlap between hardware and software is only 2.17%, so drawing the line between the two classes is quite simple with the current model. On the other hand, in order to improve the performance of security classifications, the model will need to take into account this 44% lexical overlap.

Combining the results from both tests into a final analysis it can be seen that this model has a lot of potential in classifying hardware/software failures, and is a few steps away from handling security threat classification. The near 100% accuracy on offline labeled samples suggests that the model is fitting to the data well. Observing high accuracies in the other two classes when searching for a particular class indicate that the model is good at classifying true negatives. Lastly, with the exception of the hardware class the model is out-performing the key-word based approach.

## 4 Conclusion

From the results presented in this paper it can be concluded that the weighted dictionary model is outperforming the key-word classifier, and with a little more work, the weighted dictionary model has the potential to improve performance even further. The model is successfully classifying hardware and software issues, and with the addition of a few augments it should be able to handle security threats as well.

As more samples are labeled with this semi online learning approach the model will only grow stronger. Should the accuracy ultimately approach an asymptote there still exist 3 more models that could be explored in future work.

## 5 Future Work

### 5.1 In the Event of a Semantic Drift

Should the model begin to fail due to the dictionary containing too many non-specific words, one can always remove those words from the dictionary and add them to the list of stop words. Words with weights close to 50% across all classes are not contributing much to the final classification, and could be ignored.

### 5.2 Decoupling Security and Software

The current model is susceptible to mis-classifying a class with a high lexical overlap with another class. In order to avoid these mis-classification, the model will need to be augmented in a manner that weights sentences with respect to this known overlap.

The first step would be to identify any key-words that were unique to the classification in question. Perhaps those weights could contribute more to the classification.

If there exist no unique words for that classification, then one could consider 'normalizing' the final class weight to take into account the subtle variance. If the weights only vary from 40-47%, like what was observed in this work, then that output can be mapped between 0-100%.

### 5.3 Other Semi-Supervised Models

If future work finds that this model is not preforming as well as is required, then there are other known models that will be available for use after these samples have been labeled.

The first option would be to use graphs, that tie the samples together with lexical overlap. This has been shown to work in other semantic research problems, and may show results in this work. Classifications then becomes a graph traversal problem, where labeled samples have the most energy, and closely connected samples will have some function of that energy (here the energy is the weight for the classification).

Another option would be to represent the recall data as paragraph vectors (3) and group clusters to classify. This approach is very similar to the graph method, only here one would use KNNs for classifications rather than graph traversal.

A third option would be to use word vectors in a recurrent neural network. This approach would be unique, in that rather than trying to model the sentences first before drawing connections between them, this method would try to process the sentences word by word. This could prevent the possible over-fitting in the first two options caused by memorizing on a finite list of sentences, rather than the words and their relationships with each other. The one downside of this approach is that it would require significantly more labeled samples than the previous two.

**Acknowledgments**

This research project began with Aakarsh Rao in Jerzy Rozenblit's lab at the University of Arizona, and was supported by Gregory Ditzler.

Special thanks to Sandiway Fong for meeting with the team to talk about natural language models and steering the project in the right direction.

## References

[1] Alemzadeh, H., Iyer, R.K., Kalbarczyk Z., Raman J. (2013) Analysis of Safety-Critical Computer Failures in Medical Devices. *IEEE Security & Privacy* **11**(4):14-26. Rush University Medical Center
`https://doi.org/10.1109/MSP.2013.49`

[2] Jouini, M. & Rabai, L.B.A. (2014) Classification of Security Threats in Information Systems. *Procedia Computer Science.* 32.
`https://doi.org/10.1016/j.procs.2014.05.452`

[3] Le Q. & Mikolov T. (2014) Distributed Representations of Sentences and Documents. *CoRR.*
`http://arxiv.org/abs/1405.4053`