Product Defect Mining

Final Report
CS 4624: Multimedia, Hypertext, and Information Access
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
Instructor: Dr. Edward A. Fox
Client: Xuan Zhang
May 16, 2017

Grant Golden, Jack Hall, Thomas Nguyen, Tianchen Peng, Elizabeth Villaflor, and Shuaicheng Zhang

Table of Contents

Table of Figures

Table of Tables

Executive Summary

This project is focused on customer reviews on various product defects. The goal of the project is to use machine learning algorithms to train on sets of these customer reviews in order to be able to easily identify the different defect entities within an unseen review. The identification of these entities will be beneficial to customers, product manufacturers, and governments as it will shed light on the most common defects for a certain product, as well as common defects across a class of products. Additionally, it will bring to light common resolutions for defect symptoms, including both correct and incorrect resolutions. This project also aims to make contributions to the opinion mining research community.

These goals will be accomplished by breaking the project into three main parts: data collection, data labeling, and classifier training. In the data collection phase, a web crawler will be created to pull customer reviews off of forum sites in order to create new datasets. For data labeling, datasets, both pre-existing and newly created, will be split into sentences and be assigned a defect entity based on the content of the sentence. For example, if a sentence describes a product defect, the sentence will be labeled as a symptom, and so on. Finally, in the classifier training portion of the project, machine learning algorithms will be used to classify unlabeled datasets in order to learn what types of words indicate a certain defect entity. While these are the three main aspects of the project, there are other minor phases and categories of work that will be necessary. One of these sub-phases includes designing the database tables that will be used to store the labeled datasets.

Throughout the semester the following was accomplished: the creation of a web crawler, the completion of five new datasets, the labeling of five datasets, and preliminary training results based on the linear SVC algorithm. Additionally, the new datasets and labeled datasets were uploaded into the client's preexisting database. The new datasets were collected from the Apple Community, Samsung, and Dell forum boards and include product defect reports for both hardware and software products. Based on the labeling results, and quick scans of the collected data, it was found that many defect reports contain contextual information that is not directly related to the description of either a product defect or its corresponding solution. Additionally, it was found that many reports do not include resolutions or the resolution did not actual solve the defect described. The linear SVC algorithm used for classifier training was able to accurately predict the label for a sentence about 80% of the time when training and testing occurred on similar products, i.e., two different car models. However, the accuracy was only about 60% at best when used on two completely different products, i.e., cars vs. cellphones. Overall, about 75% of the anticipated work was completed this semester. The work that was completed should provide a good foundation for continued work in the future.

1 Introduction

1.1 Objectives

The purpose of this project is to find the best machine learning algorithm for identifying the different entities, such as the symptoms and resolution of the problem, that make up a customer review or complaint about a product. The ultimate goal is to help customers, manufacturers, and governments learn of the most complained about product defects. Additionally, this project will make contributions to the ongoing research being done on opinion mining.

In order to accomplish these goals, the project will be broken up into three parts: defect entity labeling, defect entity classifier training, and the collection and testing of new datasets. Since the pieces are interdependent, there will be a set sequence for task completion, with some overlap occurring at each stage. In other words, defect entity labeling will be completed in its entirety, at least for a single dataset, before defect entity classifier training can occur. However, defect entity labeling will continue to happen on other datasets concurrently with classifier training.

1.2 User Roles and Responsibilities

There are two main users of this system: the client and the developer. The client role is defined as a customer, product manufacturer, or government official interested in learning about the common and most complained about defects for a certain product or products. In this case, the client user is not interested in the intermediate steps or inner workings of the system. The client only wants to interact with the final results, i.e., the report of most complained about defects. He or she will interact with the system by requesting results from the developer for a specific product or set of products. The client will not interact with any of the individual pieces created for this project and will not be required to understand any of the technical matter associated with the project.

The other user of this system is the developer. The developer role is defined as a researcher in the data mining field. This person will be the one actually interacting with the system, and pieces, created for this project. The developer could be a single person or a team of persons who fit the role description. The developer will need to have an understanding of the purpose and use of web crawlers, machine learning algorithms, and how to interpret the results of the machine learning algorithms. He or she will also need a coding background in order to use and customize the system; this will be discussed further in Section 2.1. As previously mentioned, the developer will actually interact with the system and will, therefore, need to know the sequence in which to use the pieces of this project in order to provide the desired results for the client.

1.3 Terminology

Classifier – a trained machine-learning algorithm that can identify different defect entities in a given defect report.

Client – a user role: a customer, product manufacturer, or government official interested in learning about the common and most complained about defects for a certain product or products

Crawler, or Web Crawler – an application that, given a website or online forum board, can extract defect reports and place them into a database

Dataset – a collection of defect reports pertaining to a particular product; in this case datasets will each be made up of 1000 defect reports

Defect, or Product Defect – an issue or complaint tied to a certain product, e.g., the home button not working on an iPhone

Defect Entity, or Entity – a piece that makes up a defect report; the two entities that will be focused on during this project are the symptom and the resolution

Defect Report, or Report – 1-3 sentences provided by a product user that outlines a product defect; during this project these reports will be taken from product website reviews and online forum boards

Developer, or Researcher – a user role; a researcher in the data mining field, with some sort of coding or computer science background

Labeling – the process of assigning different defect entities to individual pieces or sentences within a defect report, i.e., identifying the report's symptom and resolution

Record – another name for a defect report

Resolution – the steps taken to resolve the product defect or the cause of the product defect; e.g., the phone battery died and was replaced

Symptom – the product defect itself and/or the indicators that there is a product defect; e.g., there was a loud pop and white smoke, or the car refused to start

1.4 Statement of Functionality

This project will not yield a single, all-encompassing application that achieves the goals and objectives set forth above. Rather, at the end of the project there will be several independent pieces that can be used together in order to make positive contributions to the

opinion mining research field. The pieces that will be created are: a web crawler, labeled defect entity datasets, and trained defect entity classifiers.

The web crawler developed for this project will be used to collect new product defect datasets. The crawler will be easily customizable so that the developer can correctly grab defect reports and complaints off of online forums, regardless of the format of the online reports. This is where the developer's coding background becomes necessary, as not all websites display or record forum posts in the same way. The actual source code for the web crawler may need to be modified per website. Additionally, the crawler should automatically take all of the collected defect reports and place them in a database table created for the dataset. When the crawler is complete, both the customization and the collection of 1000 records should take no longer than half a day.

The labeled defect entity datasets will be used in two different capacities for this project. The first is in the training of the defect entity classifiers. The second is as a point of reference for future datasets, labeled by the classifiers. The labeled datasets will be completed by hand, with two or more people working on the labeling to ensure consistency and a common understanding of defect symptoms and resolutions. At the end of the semester, there should be at least four datasets of at least 1000 records completely labeled. These labeled datasets will be stored in database tables and will include the labels assigned by each person who worked on that dataset. Additionally, each labeled dataset will have summary statistics that include information contained in records on the most complained about defect and number of symptoms, resolutions, ignored records, etc.

The trained defect entity classifiers will be used to label datasets collected in the future, once this project has been completed. The classifiers will have been trained using a variety of different existing machine learning algorithms and the labeled datasets discussed above. The final classifiers will have been trained several times. It is expected that, at the end of the project, there will be several trained classifiers as different algorithms prove to be the best in different cases. For example, one algorithm may be better at picking up on symptoms and another at picking up on resolutions, resulting in two trained classifiers. There is no set minimum or maximum on the number of classifiers required at the end of the project.

1.5 Scope

This is a semester-long project. All requirements and goals should be completed within the semester timeframe. Each piece of this project has its own, individual timeline for development with built-in overlap as necessary. These individual timelines make development more manageable as well as prevent, as much as possible, slow development in one area from dragging the entire project behind schedule. Additionally, the overall project timeline and milestones were set by month so that the work was evenly spread out across the semester. See Table 1.

At the end of the first month, it is expected that the first dataset is completely labeled and development work on the crawler completed. Additionally, a quarter to half of the records from a second, pre-existing dataset should be labeled and 1000 records should have been collected by the crawler to create a new dataset. The labeled dataset should include a statistical report that reflects consistency and distribution trends for the entity class labels. During this phase, there is no expectation for the classifiers.

At the end of the second month, it is expected that training of the classifiers be well underway. During this phase the classifiers will be trained using SVM, Naïve Bayes, and other similarly complex machine learning algorithms. Although the classifiers should be well trained and refined at the end of this phase, training will continue into the next month and will not be complete. Additionally, the crawler should have collected a second new dataset. The second dataset should also be completely labeled, and labeling for a third dataset should be started during this month. It is important to note that the working time for this month is a week shorter, due to Spring Break.

At the end of the third month, the project should be complete. During this month, all labeling should be finalized, resulting in a total of four labeled datasets. The classifiers should be completely trained, included using Deep Neural Networks, and have final classification results for the four datasets. Additionally, a final report and presentation on the project should be completed during this month. It is expected that all work for this project be completed no later than May 3, 2017.

Table 1: Scope Summary

Month	Work Completed	
January		Project selected Semester plan created
February	Crawler	Development completed 1000 records for a new dataset collected
	Database	New tables to hold labels designed New tables to hold labels created in existing database
	Labeling	1000 records for first, pre-existing dataset labeled 250 – 500 records for second, pre-existing dataset labeled
	Training	Initial research completed Initial training plan completed
March	Crawler	1000 records for a second new dataset collected

	Database	First labeled dataset imported Second labeled dataset imported
	Labeling	1000 records for second, pre-existing dataset labeled 250 – 500 records for first new dataset labeled
	Training	Preliminary training completed
April	Crawler	N/A
	Database	Third labeled dataset imported Fourth labeled dataset imported
	Labeling	1000 records for first new dataset labeled 1000 records for second new dataset labeled
	Training	Final training completed
May		Project completed

1.6 USABILITY

The overarching usability goal for this project is that the system that is created is useful to both the client and the developer. On the client side, usefulness means that this project provides helpful and meaningful information on product defects. Further, it means that the results of the classification of data are presented in a way that a non-technical client can understand and use them to make informed decisions about a certain product. On the development side, usefulness means that the system, as well as the individual pieces, can be used past the end of this project for further research in data and opinion mining. Additionally, it means that the system created by the project reduces the amount of work a developer or researcher would have to do by hand. This would allow more time for the developer to look at the results of the mining and classification.

1.7 REPORT STRUCTURE

Following this introductory section are a User Manual and Developer Manual. These sections include helpful information and details for people using the system developed for this project in each role. Then, there is a section on the lessons learned throughout the semester while working on the project, that may be beneficial to students either extending this project or undertaking one of a similar nature. Finally, there are acknowledgements, references, and the appendices.

Detailed information on the requirements, design, implementation, and refinement of the different components of the project can be found in Appendices C - G. These appendices serve as a brief history of the flow and progression of the project throughout the semester. Information on how much and what work was completed each month of the project can be found in Appendix B. Additionally, Section 3.1 includes a brief overview of the project flow and structure.

2 User Manual

As mentioned in Section 1.2, a user would have no direct contact with the system produced for this project itself. Instead, a user would interact with the results produced by the system. Therefore, it is up to the developer or team of developers that are using and maintaining the system to provide the results in a format that is usable and understandable to a non-technical person.

The results a user receives from the developer should include details on the most common defects and resolutions for a product, as well as information that links defects and resolutions to one another, if possible. Then the user can use this information in whatever way he or she pleases.

3 Developer's Manual

This section of the report serves to provide information that would be helpful in extending this project in the future. Additionally, a general overview of the structure and flow of the project will also be given here.

3.1 Project Structure and Flow

At the beginning of the semester it was decided that this project would be split into two main sections: data collection and classifier training. In order to accommodate this, the team was split into two smaller teams each with a focus on a particular project section. For the first half of the project, the two teams operated independently and simultaneously. This proved to be efficient since the work done for each had minimal impact on the other. However, during the second half of the semester, the work became more interdependent and therefore, there was just general project work being done rather than having it split into either data collection or classifier training.

Throughout the project, the data collection section entailed just that. However, before actual collection could occur, the web crawler had to be developed. This took place while the entire team was split into the two smaller teams. Following the completion of the crawler development, data collection began. This required both research and data mining. The research required finding forum boards that were consistent with the data that was provided at the beginning of the semester. In other words, forum boards that were chosen for data collection included resolutions or responses to posts about defect and their symptoms. If a forum did not include resolutions, or included very few of them, they were not crawled. This decision was based on the fact that the actual classifier training needed to have examples of each type of entity in order to recognize the different characteristics of each entity.

The work included in the classifier training portion of the project was not as obvious. At the beginning of the semester, when the work for each section was independent, this section required labeling datasets that were provided by the client. If there had been no previously collected data, this portion of the project would have occurred following crawler development and initial collection, but contemporarily to collection of subsequent data sets. Later in the semester, the team members doing work in the classifier training section shifted focus to actually doing classifier training. This shift in focus required more interwoven work and responsibilities across section due to the use of labeled datasets and the possible need for new, unlabeled datasets.

Going forward, the project would probably not need to be split into two sections. Since the crawler is completely developed, minimal future work is required and only necessary if additional datasets are desired. However, at the end of this project there are several datasets that have been collected and are currently unlabeled. The maintenance of

the crawler is outlined in Appendices D and E. The labeling scripts should not need any modifications in order to be used going forward. The only change that needs to be made to the training script are the references to which datasets on which training and testing are to occur. In the future, labeling and training can happen simultaneously.

3.2 Inventory of Project Files

Web Crawler Files:

- Source Code:
  - geckodriver.log
  - __init__.py (x2)
  - items.py
  - middlewares.py
  - pipelines.py
  - settings.py
  - scrapy.cfg
  - selenium.csv
  - dell.py
  - dell.pyc
  - itunesformac.py
  - macbookair.py
  - macbookpro.py
  - newdell.py
- Collected Data:
  - itunesformac.csv
  - macbookair.csv
  - macbookpro.csv

Datasets:

- Samsung
  - Samsung-post.csv
  - Samsung-forum.csv
  - Samsung-thread.csv
  - Samsung-post_SPLIT_GG.csv
  - Samsung-post_SPLIT_TN.csv
- Apple Community
  - Apple_First_500.csv
  - Apple_First_500_SPLIT_GG_relabel.csv
  - Apple_First_500_SPLIT_TN_relabel.csv
  - itunesformac.csv
  - macbookair.csv
  - macbookpro.csv
- Chevy Cobalt

    - ChevroletCobalt2006_500count.csv
    - ChevroletCobalt2006_500count_SPLIT_GG_relabel.csv
    - ChevroletCobalt2006_500count_SPLIT_TN_relabel.csv
- Dell
    - dell.csv
- Ford Fusion
    - FordFusion2010_500count.csv
    - FordFusion2010_500count_SPLIT_GG_relabel.csv
    - FordFusion2010_500count_SPLIT_TN_relabel.csv
- Honda Accord 2003
    - HondaAccord2003_500count.csv
    - HondaAccord2003_500count_SPLIT_EV_UpdatedRelabel.csv
    - HondaAccord2003_500count_SPLIT_JH_relabel.csv
- Jeep Grand Cherokee 2011
    - JeepGrandCherokee2011_500count.csv
    - JeepGrandCherokee2011_500count_SPLIT_EV_UpdatedRelabel.csv
    - JeepGrandCherokee2011_500count_SPLIT_JH_relabel.csv
- Toyota Camry 2007
    - ToyotaCamry2007_500count.csv
    - ToyotaCamry2007_500count_SPLIT_GG_relabel.csv
    - ToyotaCamry2007_500count_SPLIT_JH_relabel.csv

The naming conventions of the above datasets are as follows: the CSV files without any suffix are the original dataset files exported from the database and files with the "_SPLIT_XX" suffixes are the CSV files that have had their sentences split and have been labeled by the person whose initials are XX. "GG" corresponds to Grant Golden, "TN" to Thomas Nguyen, "EV" to Elizabeth Villaflor, and "JH" to Jack Hall.

Scripts

- Labeling Scripts:
    - CSVRecordComparer.py
        - Used to compare all the labels of two labeled CSV files. It will give a count of each class of label for both files.
        - Run using: "python CSVRecordComparer.py labeledFile1.csv labeledFile2.csv".
    - CSVRecordComparerApple.py
        - Does the same thing as CSVRecordComparer.py except it is for data with thread ID's like the Apple dataset.
        - Run using: "python CSVRecordComparerApple.py labeledFile1.csv labeledFile2.csv".
    - CSVRecordsCounter.py
        - Used to count the number of records labeled so the person doing the labeling can keep track of their labeling progress.

        - Run using: “python CSVRecordCounter.py labeledFile.csv”
    - CSVRecordsCounterApple.py
        - Does the same thing as CSVRecordCounter.py but for data with thread ID’s like the Apple dataset
        - Run using: “python CSVRecordCounterApple.py labeledFile.csv”
    - CSVRecordsStats.py
        - Used on an individual labeled CSV file to count the different combinations of classes for a single record. For example, a record that only has sentences labeled as (s)ymptom will be added to the count of ‘s’ records. A record that has sentences labeled as both (s)ymptom and (o)ther will be added to the count of ‘so’ records.
        - Run using: “python CSVRecordsStats.py labeledFile.csv”
    - CSVRecordsStatsApple.py
    - CSVRelabeler.py
        - Used to automatically label all sentences less than 5 words as (i)gnored. It also changes any records previously labeled as (i)gnore that are over 5 words as (o)ther.
        - Run using: “python CSVRelabeler.py labelledFile.csv”
    - CSVSentenceSplitter.py
        - Used on the CSV file generated from a database export of the desired data. For each row/record, the script will split the text of the record by sentence and create a new row.
        - Run using: “python CSVSentenceSplitter.py unlabeledFile.csv”.
    - CSVSentenceSplitter-ThreadID.py
        - Does the same thing as CSVSentenceSplitter.py but for datasets that contain an additional ThreadID column.
        - Run using: “python CSVSentenceSplitter-ThreadID.py unlabeledFile.csv”
- Classifier Training Script:
    - trainingLabels.py
        - Used to perform the K-Fold Cross Validation Model testing and training on the given datasets. The datasets should be in two separate text files, one for the sentences and one for the labels. For the Model testing training method, the sentences from both datasets will all be in one file and the labels all in the other.
        - Run different tests by setting the “mode” variable in the main function
            - Mode = 1: 5 K-Fold Stratified Cross Validation. The sentences will be split into five groups (four for training, one for testing), each fold will be trained on the given labels, then be tested on the test set using Linear SVC. The Classification Report and Confusion Matrix will then be printed. This occurs for each fold.
            - Mode = 2: Testing-Training Model. This mode requires the

developer to keep track of where in the sentence and label files, the data changes from the first dataset to the other. Currently, the split is at 400 and should be replaced by whatever cutoff value is appropriate for the datasets being used. This is effectively the training phase where a training model is created using Linear SVC. All of the sentences after the cutoff are passed to the test model, where all sentences after cutoff are then tested upon using the model previously created.
    - Any other value will result in nothing happening.
- A time is kept running throughout the training and testing and following completion, the total runtime will be printed for the user to view.

4 Lessons Learned

4.1 Timeline

A detailed version of the timeline for this project is provided in Table 1. For the most part, the schedule set forth at the beginning of the semester was easy to adhere to. However, beginning in mid-March, the schedule became restrictive and left very little room for assigned tasks to take longer than the allotted two weeks. Therefore, problems arose, discussed below in Section 4.2, and caused the project to fall behind. Additionally, these problems resulted in accomplishing fewer things than originally anticipated.

4.2 Problems and Solutions

Although there were a few speed bumps that arose during data collection, they were easy to handle and find workarounds for. These issues mostly concerned forums being large and overwhelming for a human reader (containing a couple hundred of threads) but too small to comprise a usable dataset for the purposes of this project (1000+ threads), A detailed description of such issues is given in Appendix G: Crawler.

Similarly, the issues that arose during the labeling phase were relatively simple. The issues in this case were a direct result of overconfidence in the quality of the content of the datasets both provided and collected. Namely, many of the provided "resolutions" were either incorrect or not obvious solutions to a problem. Additionally, many sentences contained contextual information that was not pertinent to the product defect. A detailed description of these issues and their corresponding solutions can be found in Appendix G: Labeling. While these issues were small and easy to handle, recognizing that they were issues and coming up with a plan to solve them ended up being very time consuming.

The largest and most troublesome issue, however, was a general lack of knowledge and experience with machine learning and classifier training. Much research and trial and error experimentation had to occur before serious training could take place. Even once actual classifier training and testing began, several mistakes were made that required results to be thrown out and the process started over. These setbacks began to pile up and resulted with only one machine learning algorithm being used for classifier training. This decision was based on the group's desire to produce one set of quality results and classifications, rather than try to produce several mediocre sets of results and classifications. Should this project be continued, some experience with machine learning and/or classifier training is strongly encouraged. While the client was understanding and more than willing to help teach, it was disappointing to not be able to produce all of the expected results.

4.3 Future Plans

If this project were extended, the main plan would be to continue to test different classification algorithms. Some of the algorithms that could be explored are Naïve Bayes, SVM, and a Deep Neural Network. Additionally, further labeling of already labeled datasets might result in a more even distribution of labels. The effect of a more even distribution of labels on the classification results would be interesting to see. Alternately, the crawler could be used to pull data from a forum where the defect reports do have a more even distribution. Another interesting thing to explore in the future would be the effect, and effectiveness, of having subclasses or sub-labels for the different types of resolutions.

Acknowledgements

This project was supported by Xuan Zhang. We'd like to thank Xuan for all of his help, insight, and guidance, without which we would not have been able to accomplish the work we completed this semester.

References

[1] Apache, "Nutch," http://nutch.apache.org/ date accessed: February 3, 2017

[2] Internet Archive, "Heritrix," http://crawler.archive.org/index.html date accessed: February 3, 2017

[3] Scrapinghub, "Scrapy," https://scrapy.org/. date accessed: February 6, 2017

[4] Oracle, "MySQL," https://dev.mysql.com/doc/. date accessed: February 6, 2017

[5] A. Becker, "HeidiSQL," https://www.heidisql.com/. date accessed: February 10, 2017

[6] "scikit-learn," http://scikit-learn.org/stable/documentation.html. date accessed: March 4, 2017

[7] Aldawud, O. *Software Design Specification Template*. Illinois Institute of Technology. date accessed: February 18, 2017

[8] T. Berezin, "General Principles in Writing a Requirements Document," Adelphi University, 1999. date accessed: February 18, 2017

[9] S. Pal “Sentence Genre Classification using Scikit-Learn Linear SVC,” ed: http://sujitpal.blogspot.com/2013/08/sentence-genre-classification-using.html date accessed: March 7, 2017.

Appendix A: Team Member Roles and Responsibilities

Table 2: Team Member Roles and Responsibilities

Team Member	Role	Responsibilities
Elizabeth Villaflor	Team Lead	• Main contact between group and client and professor • Keep track of upcoming deadlines • Record minutes at meetings • Assist in dataset labeling
Grant Golden	Labeling Lead	• Oversee dataset labeling • Delegate labeling tasks to other team members • Compile statistics on dataset labels • Assist in classifier training research
Jack Hall	Training Lead	• Research and develop a plan for classifier training • Compile research from other team members • Oversee classifier training • Assist in dataset labeling
Thomas Nguyen	Database Lead	• Design and create table schemas for storing labels in database • Import labeled data into database • Assist in dataset labeling
Tianchen Peng	Crawler Development Lead	• Oversee web crawler development • Assist in collecting new datasets • Assist in classifier training research
Shauicheng Zhang	Data Collection Lead	• Compile list of candidate websites to crawl • Oversee collection of new datasets • Assist in web crawler development

Table 2 outlines the roles and responsibilities for each team member throughout the semester. Each team member was assigned to be the lead for a different aspect of the project. Additionally, each member was given the responsibility of being a main assistant for a different aspect of the project from the one for which they are the lead. These role assignments were determined based on each team members' specific interests within the project as well as their personal strengths and skills.

Appendix B: Past Progress Reports

COMPLETED WORK: FEBRUARY

At this point, the project is at the first major milestone in terms of work that should be completed. Based on the project timeline set forth at the beginning of the semester, the following should be finished by the end of February:

- 1000 records from the vehicle defect dataset labeled
- 250-500 records from iPhone dataset labeled
- 1000 records collected to create a new dataset
- Crawler completed
- Generated report on labeling consistency/class distribution stats

Additionally, although not explicitly stated, the database should be set up according to the agreed upon design in order to support the above work.

So far, about half of the vehicle defect dataset has been labeled. Initially, labeling was slow due to figuring out how to go about labeling and parsing the data into sentences to be labeled. However, now that all of the vehicle defect reports have been separated out into vehicle-specific sets and parsed into sentences, completing the second half of labeling should be done easily by the end of the month. Moreover, having figured out the setup required prior to labeling, the first chunk of labels for the second dataset should also be easily completed for this milestone. Once the labels are completed, a report including label statistics can be quickly generated using built-in functions in Excel.

Additionally, the crawler has been completely developed and is currently undergoing testing on websites to see how quickly and well it can collect data. A candidate list of websites to crawl is currently being compiled. Following approval of the websites by the client, a new dataset will be collected.

The database design is finalized and tables are being built. Currently, the database has tables for, and stores the sentence data related to, each of the five vehicle subsets. A test label table is in the process of being built and populated with the completed labeling results for the 2010 Ford Fusion defect reports.

Finally, initial research into machine learning algorithms is being conducted. Although research is not an explicit goal for this month's milestones, it is necessary so that the work for the next month milestones, which are training heavy, can be completed on time. Being proactive during the final stretch of this month is especially important when looking ahead at the loss of a week in March due to Spring Break.

COMPLETED WORK: MARCH

At this point, the project is at its midpoint and three of the six major milestones have been completed. Significant progress has been made in all aspects of the project and, for the most part, the project as a whole is right on schedule. Based on the timeline established at the beginning of the semester, the following should be complete at this time:

- Have 1000 records from provided dataset labeled
- 250-500 records from second dataset labeled
- 1000 records collected to create a new dataset
- Crawler completed
- Have a report on labeling consistency/class distribution stats
- Have preliminary classifications completed
- Start evaluation of training (starting with SVM, Naive Bayes, etc.)

Based on the above tasks, the labeling portion of the project is well underway. Not only have 1000 records from one dataset and 250 records from a second dataset been labeled, but also both sets of labeled records have had statistics produced for them. These statistics include the number and percentage of sentences that fall into each label category. However, there have been more challenges than expected. For example, an initial round of labeling proved the need for a new label, "mixed", to be created since some sentences included both a symptom and a resolution. The creation of this new label required labeling to be restarted to accommodate the new label and any sentences that may have fallen into that classification. Additionally, as seen in Table 3, some of the datasets have a very low percentage of matching labels, i.e., many of the sentences in the dataset were given different labels by two team members. Some of the reasons for these discrepancies are a result of misunderstanding what the "other" and "ignore" labels entailed. So, while the desired number of labels have been completed, some re-labeling will be necessary prior to adding the labeled sentences to the database. Although there is no explicit mention of progress pertaining to the database in the project timeline, the tables have been created and are ready to store labeled sentences.

Table 3: Label Matching Percentages by Dataset

Dataset Name	% Matching
Chevrolet Cobalt	64
Ford Fusion	83
Toyota Camry	84
Jeep Cherokee	57.3
Honda Accord	71.8
iOS	36.3

In terms of the tasks related to the crawler, the development has been completed but there have been several complications related to pulling defect reports in order to create a

new dataset. So far, the crawler has been able to collect over 1000 records, however, these records cannot be used to create a single, new dataset. This is because the crawler pulled 400 defect reports each on the MacBook Air, MacBook Pro, and iTunes for Mac from the associated Apple Community sites. But, since the records deal with different products, which are not all in the same class of product, they must each be standalone datasets. Unfortunately, Apple Communities appears to limit the number of posts on a product forum to 400, so these datasets cannot even be expanded to the required 1000 records. So, alternate websites to crawl are currently being researched. However, the successful collection of over 1000 records proves that the crawler does, in fact, work as expected.

The portion of the project that is furthest behind is the actual classifier training. This is due, in part, to many of the obstacles faced during labeling. Since labeling has been more time intensive than expected, research into classifier training was delayed. So, while preliminary training has not yet happened, it is slated to start at the beginning of next week.

Completed Work: April

At this point, the project is about a month out from completion. While some aspects of the project are slightly behind schedule, there should be no issue with completing the work ahead. The web crawler was able to collect enough records to create two more new datasets, each including about 1000 records. This exceeds the expected number of datasets that were to be created this semester by one. As with last month, the database is poised for data import and is on standby until the dataset labeling is complete.

While the dataset labeling has taken significantly more time than expected, this portion of the project is beginning to taper off towards completion. While there are still 1.5 – 2 whole datasets to label, a strict set of rules and a process to follow should make the work quick and easy to complete. However, some preliminary results on the distribution of defect entities across defect reports have been generated. These stats show that the majority of records contain at least one “symptom” statement and at least one “other” sentence. Additionally, these stats show that very few records include both a “symptom” and a “resolution” sentence and a fair number of records include just “symptom”s.

The majority of the focus now, and for the rest of the semester, lies in the classifier training. Getting started has been slow due to the learning curve associated with having no prior machine learning or classifier training experience. However, work on this phase has begun and is already showing promising results. The following section, 5.6, further outlines where exactly the project stands versus where it is going and should end up by the end of the semester.

Plans for end of April – May

With several thousand records collected by the crawler, there are very few future plans for this portion of the project. While there will be some continued work and research into whether or not the Apple Community's database can be directly accessed, this will not be a main concern. Instead, the focus will shift toward wrapping up labeling and completing the project as a whole.

In terms of labeling, the only future plans include hand-labeling the newly collected datasets so that the results of the trained classifiers can be compared to expected values. Additionally, the completed labeled datasets will be imported into the database, with the goal of all of the datasets being stored in the database by the end of the month.

Going forward, the bulk of the focus will be on classifier training. Currently, most of the work is going into making all of the training code more robust. At the moment it relies heavily on hard-coding file names as well as the necessity to run two different scripts. Hopefully, this process can be streamlined before the project is completed. Another goal for this portion of the project for the rest of the semester is the addition of the ability to train a data set, and then use a second dataset as the test set into the training code. This would expand upon using K-Fold cross validation on a single set of labeled sentences.

Appendix C: Requirements

Business Process

There is no current business process for the system being built during this project. The only access customers and manufacturers have to information concerning product defects is on online forum boards. However, these boards don't necessarily allow the person viewing reviews to sort based on problem or to see any statistics on which defects are most common. Instead, these boards are simply organized by product.

The introduction of this system would create a business process that would allow researchers in the data mining field to grab reviews off of these forum boards, create statistics on defects, and present the information to the producer of the product. Then the product manufacturer would have the option to share the defect statistics with potential consumers either via the statistics themselves or through filters on forum boards. If the manufacturer did not choose to share the information, they would still be able to use it internally. This information would be helpful to product producers because it would allow them to see what issues are common so that they could focus on which aspects of the product need to be redesigned or otherwise reworked.

Interactions with Other Systems

As far as the scope of this project, the system created will not be required to interact with other systems. However, the individual pieces of this project may be used with other systems in the future.

Replacement of Legacy Systems

The system created for this project will not be replacing any current system.

Production Rollout Considerations

This project has several milestones built into its timeline. These milestones will allow for a continuous cycle of testing and refining of project pieces throughout the semester. Therefore, the project should be complete and ready for use in production without any further modification to the system. It is expected that by the end of the semester the system will include: a completed web crawler that has been tested and used to collect at least two new datasets of 1000 records each, four labeled datasets, trained classifiers to use for labeling future datasets, and statistics on which machine learning algorithms are best suited for training the classifiers.

PERFORMANCE

There are no specific performance requirements for the labeling aspect of the project. As previously mentioned, the expected performance of the web crawler is that both customization and collection of at least 1000 records should take no longer than half a day. The expected classifier performance is both an 80% accuracy and precision score when tested on an unlabeled dataset.

Appendix D: Design

CRAWLER

The web crawler will be developed from scratch using a pre-existing web framework. Using an existing, open source framework will allow the crawler to be customized for the tasks associated with this project. Additionally, it will reduce the development time and learning curve associated with developing a web crawler, something no member of the team has experience doing. Some of the frameworks that are currently being considered are Apache Nutch[1] and Heritrix[2]. However, a final framework has not been determined.

Two major design considerations for the crawler deal with the websites that will be mined themselves. The crawler will have to be created in a way so that it is flexible and can easily be changed to handle the various formats of forum boards and other websites that will be mined for data. Moreover, the crawler will also have to be developed to work around any websites that have protections built in to prevent them from being crawled. This will probably be accomplished by having a delay built into the crawler's requests to a page.

DATABASE

The database that will be used for this project has already been created and currently stores two sets of product defect report data. These datasets are simply stored as full complaint reports, each with a unique ID. However, for the purposes of this project, these reports will have to be spilt into individual sentences. Therefore, the design considerations for the database portion include how to store these sentences, with their associated labels, and maintain a link to the original report. This means that new tables and schemas will have to be determined and created in the existing database. The new schema will need to handle complex links between several tables without requiring several intermediate tables, since it will have to be duplicated across at least four datasets that each contain 1000 records. Additionally, the schema should be intuitive so that it is still useful and usable following this project's completion.

LABELING

The labeling process is one of the most important aspects of this project, especially in the early stages. This is because the labeled sentences will be used during classifier training later in the project. As mentioned above, individual records will be split up into multiple sentences. These sentences will each be assigned a label associated with a different defect entity. The possible label types are: 's' for symptom, 'r' for resolution, 'm' for mixed, 'o' for other, and 'i' for ignore. Table 4 provides further explanation of each label.

Table 4: Label Descriptions

Label	Defect Entity	Description
s	Symptom	Sentence describes a product defect
r	Resolution	Sentence describes a solution for a product defect (may or may not be correct)
m	Mixed	Sentence includes a product defect and an associated resolution
o	Other	Sentence is at least six words long and does not fall into the symptom, resolution, or mixed category
i	Ignore	Sentence is five words or fewer

During the labeling process, there will be two team members assigning labels to each sentence. Then, statistics will be run to determine how similar the labels for the same sentences are. Once the level of discrepancy is determined, rules for each label can be set and appropriate sentences can be relabeled. This means that, for at least the first dataset, there will be multiple rounds of labeling before the labels for a particular dataset are finalized. However, the creation of labeling rules should be applicable to multiple datasets, therefore reducing the amount of relabeling that occurs for the second, third, and fourth datasets. Using this procedure will ensure consistency in labeling across datasets and team members. Having consistent and correct labels is extremely important since they are what will be used during classifier training in the later sections of the project.

CLASSIFIER TRAINING

Since the training of the classifiers is reliant on a completely labeled dataset, training cannot begin until at least one round of labeling is finished. Therefore, the first stage of classifier training is research. The necessary research includes looking into which machine-learning algorithms are the best candidates for producing the desired results, if the candidate algorithms are available for use in existing libraries, and how to execute and test these algorithms. So far, the candidate algorithms include: SVM, Naive Bayes, and Deep Neural Networks. All of these algorithms are already implemented in existing libraries and readily available for use. However, this list of algorithms is neither complete nor final and more research still needs to be conducted.

Once the list of algorithms to use in training has been finalized and a complete dataset is labeled, the actual classifier training will take place. The first half of training will be done using the less complex algorithms like SVM and Naive Bayes. During the second half of training, once it is understood how to execute and test different algorithms, more complex algorithms will be used for training. At this point, the algorithm that will be used for training classifiers during the second half will be the Deep Neural Network algorithm.

Appendix E: Implementation

CRAWLER

In order to get reviews from websites such as technical forums, a web crawler using the Scrapy[3] framework was created. After the decision regarding which websites will be crawled for new datasets has been finalized, the website's HTML and pattern of the website will be analyzed. Then, the crawler code can be adjusted to retrieve the review information from the forums. Since some websites have anti-crawler settings, such as detecting the time between each request and preventing requests that occur too frequently, our crawler is set to have some delay between requests. This delay will allow the crawler to pretend to be a human user browsing the site rather than a crawler collecting information. A pipeline is also set up to store the obtained information from the website, directly into a MySQL[4] database.

DATABASE

After obtaining a dataset from crawling a new source, the records will be kept inside a database with certain schemas. This database includes two datasets that were collected prior to the beginning of the project. These datasets include:

- More than 1 million vehicle complaints from NHTSA
- About 1,700 iPhone complaints from Apple Communities

In order to make the vehicle complaint dataset more manageable, smaller datasets were extracted based on specific vehicle makes and models. These smaller datasets were exported into a complaint table with two fields, the unique complaint ID and the complaint description, for the 2010 Ford Fusion, 2006 Chevrolet Cobalt, 2011 Jeep Grand Cherokee, 2007 Toyota Camry, and 2003 Honda Accord.

Each dataset is comprised of defect reports, most of which span several sentences. In order to correctly identify and label the different entity sentences, each report was split into its individual sentences but still clustered by a single ID (CMPLID). Figure 1 shows an example of what the defect reports look like when split into sentences. Note that none of the reports in this sample contained multiple sentences and, therefore, each is row is tied to a unique CMPLID.

CMPLID	CDESCR
722283	TL*THE CONTACT OWNS A 2010 FORD FUSION SPORT. THE CONTACT STATED THAT THE WINDSHIELD CONTAINED A STRESS CRACK DUE TO THE HEAT. THE VEHICLE WAS PURCHASED ON APRIL 13, 2009. THE VEHICLE IS CURRENTLY AT THE DEAL
725476	CAR WAS PURCHASED NEW CAR IS 2010 FUSION HYBRID. NOTICED CRACK IN WINDSHIELD STARTING UNDER REARVIEW MIRROR DO WEST FOR APPROX 3 INCHES. CRACK HASN'T GOTTEN BIGGER. DIDN'T NOTICE AT TIME OF DELIVERY. *TR
726797	I BOUGHT A 2010 FORD FUSION HYBRID ON MAY 23, 2009 FROM CROWN FORD OF FAYETTEVILLE, NC. ON MAY 27, WHILE DRIVING DOWN THE ROAD, THE BREAKS SUDDENLY HIT THE FLOOR AND THERE WAS MINIMAL RESISTANCE. I EVENTUA
736088	THE HOOD ON MY 2010 FORD FUSION HYBRID UNLATCHES ITSELF AT INTERSTATE HIGHWAY SPEED. THE SAFETY CATCH HAS SUCCESSFULLY RESTRAINED THE HOOD FROM FLYING OPEN COMPLETELY. THERE HAVE BEEN FOUR INCIDENTS WITH T
739090	TROUBLE STARTING LIKE THE STARTER IS DEFECTIVE, BUT THAT DOESN'T SEEM TO BE THE PROBLEM. AUTOMATIC TRANSMISSION THAT JERKS WHEN SHIFTING. *TR
739091	TROUBLE STARTING LIKE THE STARTER IS DEFECTIVE, BUT THAT DOESN'T SEEM TO BE THE PROBLEM. AUTOMATIC TRANSMISSION THAT JERKS WHEN SHIFTING. *TR
740881	TL*THE CONTACT OWNS A 2010 FORD FUSION. WHEN ACCELERATING UPHILL, THE VEHICLE ROLLED BACKWARDS. AS A CONSEQUENCE, HIS RISK FOR A VEHICLE CRASH WAS INCREASED. THE VEHICLE HAS NOT BEEN INSPECTED TO DETERMINE T
742993	THE HEAD REST ON MY NEW 2010 FORD FOCUS (NOT A FUSION AS NOTED) PROTRUDES FORWARD BEYOND THE TOP OF THE SEAT CAUSING MY HEAD AND NECK TO BE PUSHED FORWARD INTO AN UNNATURAL POSITION WITH MY SPINE. THIS,
750246	MY COMPLAINT IS NOT SPECIFIC TO ANY PARTICULAR AUTOMOBILE, BUT INSTEAD CONCERNS THE FRONT SEAT HEADRESTS IN MANY LATE-MODEL VEHICLES. THESE HEADRESTS ARE ANGLED TOO FAR FORWARD AND ARE CAUSING DRIVERS AN

Figure 1: Sample Dataset Sentences

The 2010 Ford Fusion complaint table was used to parse the defect reports into readable sentences. A sentence table for the Ford Fusion data was created with the following fields:

- A unique integer: sentenceID, as the primary key
- Another integer for the original complaint ID: CMPLID, as reference
- A variable character field (varchar) of a single sentence: SENTENCE

This table is used to store records of multiple sentences pertaining to a specific complaint ID.

Following the sentence table, a label table is created with the following fields:

- An unique integer: labelID, as the primary key
- An integer: sentenceID, as a reference
- A varchar: TYPE, for the entity type (symptom, resolution, ignore, other)
- A varchar: TAGGER, to indicate who selected the entity type

This label table will be used to compare entity types chosen by the taggers to evaluate any discrepancies; additionally, the labels will also be used to help train classifiers.

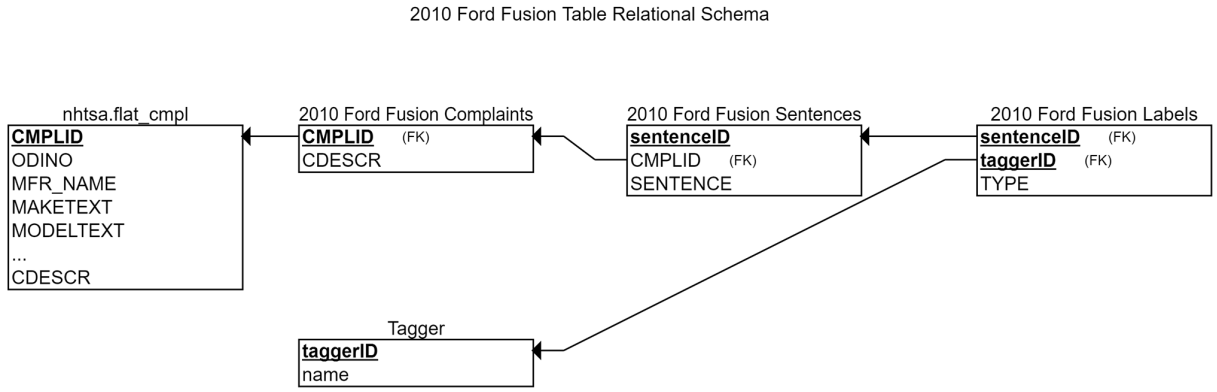

Figure 2: 2010 Ford Fusion Table Relational Schema

This schema will be followed for all of the other vehicle datasets as they are processed. Additionally, similar schemas will be followed for the three other datasets later in the project.

LABELING

Once a data set is collected in the database, it is ready to be labeled. The first step of this process is to query the relevant fields (the complaint ID and complaint description, in this case) with an SQL statement like:

```
SELECT CMPLID, CDESCR FROM flat_cmpl WHERE YEARTXT = 2010 AND MODELTXT =
'FUSION' LIMIT 500;
```

The expectation for the first phase of labeling is that each of the five vehicle-specific datasets have 200 records labeled. The above query would yield the desired 500 records for a 2010 Ford Fusion. The 200 record limit for labeling was chosen after initially pulling 500 records, skimming the records, and finding a large number of duplicate complaints.

Once the query is executed and the data is returned, the data is exported to an Excel file so that it can be easily labeled. This export can be done with many different instances of SQL client software; for this project HeidiSQL[5] was chosen since it directly transfers the query results to a CSV file. At this point, the CSV file will resemble the returned data from the database query with one column for complaint ID and one column for complaint description. Since the labeling needs to be done for each sentence, not for each complaint, each complaint description needs to be split into individual sentences. Doing so creates a new row for each sentence. Rather than doing the splitting by hand, this process is done with a Python script that parses the CSV file row by row and creates a new row for each sentence. The script also adds two new columns: type and tagger, which correspond to columns in the label tables as described by the database implementation above. The possible labels are described in Appendix D. The tagger column gives the name of the person who did the tagging/labeling. This column is necessary for comparing the tags/labels for two different people's labeled data. Finally, the script outputs the resulting spreadsheet as a new CSV file that is ready for manual labeling.

Manual labeling involves going row by row and assigning a label to each sentence. The manual labeling is done independently by at least two people in order to ensure that there is a consensus on what it means for a sentence to be a symptom, resolution, ignored data, or other. This process will be used throughout the manual-labeling phase for all four datasets. At this point, since each record is broken up into multiple sentences and the number of sentences per defect report is inconsistent, it is difficult to track the number of records that have been labeled. Therefore, another Python script was created to count the number of labeled records. Tracking the number of records that have been labeled is important since manual labeling will be time consuming and four 1000-record datasets have to be completed in a single semester time frame, with other work to be done as well. Moreover, it is important to track the number of records labeled for the vehicle defect reports, since it is made up of five smaller datasets, each of which needs an equal number

of reports labeled.

Classifier Training

As mentioned in Appendix D, classifier training will occur in two phases. The first phase will include simpler algorithms, while the second will include more complex ones. In order to run these algorithms and actually train classifiers, an open source Python library called scikit-learn[4] will be used. Once again, the use of an open source library will reduce the learning curve associated with running machine learning algorithms and training classifiers. Any reduction in learning curve possible is appreciated, and necessary, given the short timeline of the project in addition to the lack of previous experience in machine learning from team members. Additionally, existing libraries already include popular algorithms, meaning that the team would not have to figure out how to write the algorithms from scratch. Scikit-learn[6] was the library chosen since it was able to perform both simple and complex classification algorithms.

As far as running the actual algorithms and training classifiers goes, it is not suggested that the algorithms are run on whole datasets at a time. Instead, the datasets should be partitioned and training should occur on each partition. Moreover, during training, each algorithm should be run on the same dataset multiple times. In order to account for both the partitions and number of iterations, k-folds cross validation will be used. This type of model selection is also built into the scikit-learn library. K-folds cross validation will run each algorithm k times and break the given dataset into k randomly partitioned subsets. For this project, k will be 5. Using this built in function and the same value of k each time will allow training to be consistent across datasets.

Appendix F: Prototype

CRAWLER

The crawler prototype was built based on the design and implementation described in Appendices D and E of this report. After using the prototype to scrape data from the Dell Community forum, only slight modifications were needed in order to use the crawler to mine data from other sites. In other words, no major refinements were necessary for this portion of the project in order to use it going forward.

Since each website's page source is slightly different, slight modifications to the crawler are required before retrieving data. These modifications are done on the pipeline aspect and callback method used when the crawler encounters a page. The pipeline modification is done by setting the MySQL port username and password and turning on the pipeline setting built into the Scrapy framework. This setting allows the data pulled from a website to be automatically stored in the associated database. For different community forums, Scrapy selectors are used to select the desired data, such as the title or the date, and store them as a variable. Since every community forum is different, we can filter the information by checking the HTML tags.

DATABASE

There was no prototype associated with this aspect of the project.

LABELING

The prototype associated with the labeling portion of the project was the first round of labeling performed by each member assigned to data labeling. As can be seen in Table 3 (in Appendix B), some datasets had a wide range of discrepancy between labels assigned by two different team members. This is the reason why the first round of labeling is considered the prototype. The results were examined and discussed and changes to the labeling process were made before continuing to label the remaining datasets. The changes that were made and the subsequent results are discussed in Appendix G.

CLASSIFIER TRAINING

The prototype associated with the classifier training portion of the project was focused around using existing documentation and other datasets. Since the team was new to scikit-learn, training on project datasets could not occur right away. Instead, a fair amount of time was dedicated to reading tutorials on machine learning and classifier training, as well as using the examples found on the scikit-learn website. After the team was fairly comfortable with the examples, sample code snippets and datasets, provided by the scikit-learn library, were used for training and basic text classification. This as a

prototype allowed for a better understanding of the effectiveness of different machine learning algorithms, proper ways to classify text, and a great starting point into training project-specific data.

Appendix G: Refinement and Testing

CRAWLER

As mentioned in Appendix F: Crawler, aside from the minor changes that need to be made when mining a new site, there was no refinement required for the crawler prototype.

Testing on the crawler consisted of using it to collect customer reviews from the Apple Community forums, Samsung Community forums, and Dell Community forums. The crawler was able to grab all the posts from one thread and store them in the database as individual records with the same threadID but a different postID. This was accomplished for all three of the community forums mentioned above.

One issue that occurred during this testing phase had to do with pulling data off of the Apple Community forums. Unfortunately, the threads on the Apple community are only cached for 400 records for the speed of the website; therefore, more data has to be requested by using the same method that Apple community uses to get data from their database. However, the JavaScript source code that is used by Apple to retrieve data from their database could not be found. Therefore, no more than 400 records could be collected at a time. In order to handle this and build a larger dataset, the forum was crawled once a week and duplicate entries, i.e., ones that persisted from the previous week but were otherwise unchanged, were deleted from the dataset.

A similar issue occurred when crawling the Alienware forum, i.e., the crawler could only gather defect reports displayed on the initial page. However, changing pages within the forum manually caused the site to pull new data, allowing for more records to be collected. An auto web testing tool called Selenium was then used to simulate a user actually opening a web browser and paging through the forum. Since Selenium was collecting all of the page links at once, rather than one at a time like with Scrapy, more data was able to be collected. However, after observing the effects of requesting 500 pages at a time, it became clear that when more pages are requested, the data retrieval is slower. Therefore, a fixed wait time is set in order for all of the data to load completely.

Another issue that was encountered during testing involved storing the records. Sometimes the length of a post's title was longer than the database field was set to hold. Luckily, this was an easy issue to solve. Simply changing the VARCHAR type to TEXT type allowed for the longer post titles to be stored.

DATABASE

Following the completion of the first round of labeling, the Ford Fusion dataset was put into the database to test whether or not the schema design was a good fit. The import was successful and no changes were required in order to accommodate the labeled data. At the end of the semester, once all labels are finalized, one mass import of all of the datasets

will occur.

LABELING

As mentioned in Appendix F: Labeling and in Table 3, some of the datasets had a wide range of discrepancies across the labels assigned to sentences. Therefore, the labeling results were analyzed and discussion of how to refine the labeling rules and process took place. During this time it was found that the majority of these discrepancies were a result of either misunderstanding or misinterpretation of the original labeling rules. Additionally, there was no evidence found during either data analysis or discussion that any of the discrepancies were a result of flaws in the labeling process. Therefore, no changes were made to the process. There were, however, minor modifications made to the labeling rules.

The two labels that caused the most issues were "resolution" and "ignore." For resolution, the issue had to do with whether or not the label included "failed" resolutions, i.e., the dealer replaced the engine but the problem persisted. Since the original labeling rules state that resolutions may or may not fix the actual problem, it was decided that "failed" resolutions would also be included under the resolution umbrella. Additionally, the team made note that if either time permitted or this project was continued into another semester, it might be helpful to break down resolution into separate labels for correct resolution, failed resolution, and possible resolution. Doing so would allow for more accurate classifier training and symptom/resolution identification.

The main issue that was associated with the ignore label was whether or not it included nonrelevant information. For example, many of the reviews included contextual sentences, e.g., I bought my car on 9/3/2011, which did not explicitly state a symptom or resolution. Some team members took the description of the ignore label, stated in Table 4, simply as a starting point and included the contextual information with sentences that should be ignored. Other members, however, took the description at face value and only labeled short sentences as "ignore". These team members labeled the contextual sentences as "other". Following a discussion with the client, the team decided to keep "ignore" for sentences that included five words or fewer and add contextual sentences as part of "other". The ignore label was, however, expanded to be used for sentences that were incorrectly split by the Python script. These sentences usually included a date or ellipsis that caused the sentence to be split in the middle and spanned multiple sentences. In these cases, the first sentence was labeled according to the content of the entire sentence and the subsequence sentence pieces were labeled "ignore". Table 5 includes a summary of the revised labeling rules.

Table 5: Revised Labeling Descriptions

Label	Defect Entity	Description
s	Symptom	Sentence describes a product defect
r	Resolution	Sentence describes a solution for a product defect (includes correct, incorrect, and possible resolutions)
m	Mixed	Sentence includes a product defect and an associated resolution
o	Other	Sentence is at least six words long and does not fall into the symptom, resolution, or mixed category
i	Ignore	Sentence is five words or fewer OR additional sentences created as a result of an incorrect split

Following the discussion of the results of the first labeling and the revision of the labeling rules, the datasets were relabeled according to the new rules. The results of this relabeling are included in Table 6 below. Across the board, the percentage of matching labels for sentences increased following the relabeling.

Table 6: Label Matching Percentages by Dataset Following Relabeling

	% Matching	
Dataset Name	**Initial Labeling**	**After Relabeling**
Chevrolet Cobalt	64	90
Ford Fusion	83	87.9
Toyota Camry	84	89.3
Jeep Cherokee	57.3	78
Honda Accord	71.8	79.7
iOS	36.3	77.8

There was no testing necessary for the labeling portion of the project.

CLASSIFIER TRAINING

Classifier training is dependent on the full completion of dataset labeling. Since labeling is not currently complete, classifier training is currently being done on a small scale using one completed set and performing simple tests. Later, the training will be expanded to include the other datasets.

The dataset being used is classified using a bag-of-words approach. Preliminary testing on this set included 402 sentences with 1037 features. To train and test the set, a Stratified K-fold with 5 folds and randomization and shuffling enabled was used. A Linear Support Vector Machine algorithm was also used to train the data. All of the libraries necessary to do this are contained in scikit-learn. Overall, the dataset results in an average accuracy of 75-80%. This is a promising accuracy range given that the dataset's hand-labeled counterpart has 89% agreement between team member labels.

After completing the configuration of the label training script (e.g., setting the number of folds, where to get data from, etc.), results could be produced. The 2007 Toyota Camry dataset was randomly selected to use to train and test on, using cross validation. The overall runtime of the script took 0.36 seconds. Most of this time was spent printing out the results. Scikit-learn has an excellent function that shows the breakdown of statistics for each label class. This allows for more statistical information to be extracted from the training and testing. Below are the tables showing an average of precision, recall, and F1-score over the 5 folds for the different labels (Table 7) as well as an average of the 5 confusion matrices for each fold (Table 8). Note that the class with little support greatly suffered. This data set was densely populated with "other" and "symptom" labels and, as a result, it was tough to correctly predict "resolution" labels because of the limited number of resolution sentences available in the labeled dataset.

Table 7: Average of Classification Report – Toyota Camry

Label	Precision	Recall	F1-Score	Support
Other	0.79	0.86	0.82	41
Symptom	0.83	0.81	0.82	37
Resolution	0	0	0	3
Average	0.78	0.80	0.79	

Table 8: Average of Confusion Matrices – Toyota Camry

	Other	Symptom	Resolution
Other	35	6	0
Symptom	7	31	0
Resolution	2	1	0

The transfer learning, or inductive learning, method for testing and training was used for this project. The idea was that one dataset could be trained on and then using this training, a different dataset could be tested. The 2007 Toyota Camry was used for training and the dataset used for testing was 2006 Chevrolet Cobalt. This ran in 0.26 seconds. Below are the tables showing the precision, recall, and F1-score for testing upon the Cobalt dataset after training a model using the Camry dataset (Table 9) as well as the confusion matrix (Table 10). Once again, the labeled classes with less support suffered. In this case, those were the "resolution" and "mixed" labels. Theoretically, with a larger sample from these two, better results could be expected.

Table 9: Classification Report – Camry vs. Cobalt

Label	Precision	Recall	F1-Score	Support
Other	0.79	0.85	0.82	206
Symptom	0.79	.079	0.79	180
Resolution	0	0	0	13
Mixed	0	0	0	1
Average	0.76	0.79	0.78	

Table 10: Confusion Matrix – Camry vs. Cobalt

	Other	Symptom	Resolution	Mixed
Other	175	31	0	0
Solution	38	142	0	0
Resolution	8	5	0	0
Mixed	0	1	0	0

Based on the tables, it can be seen that both of these models produce similar scores. This is somewhat expected because they are both car defect datasets. This allows the trained and tested sentences to be similar, even though they came from different datasets. Things get more interesting when using datasets for two completely different products. To show this, training was done using the Samsung dataset and testing upon the 2006 Toyota Camry dataset. It took 0.49 seconds for the script to execute. Below are tables that show the classification report (Table 12) and confusion matrix (Table 11). It is no surprise that the results are not as good. There are two reasons for this. First, the two datasets share little in common, so it is much more difficult to predict the label based on the actual sentence content. Second, the Samsung set was mainly comprised of "other" sentences (over 1000) compared to "resolution", "symptom", and "mixed" (all combined to 200). This causes the predictions to skew towards "other" which explains the high recall for "other" but low recall for the rest of the labels.

Table 11: Classification Report – Samsung vs. Camry

Label	Precision	Recall	F1-Score	Support
Other	0.52	1	0.68	206
Symptom	1	0.01	0.02	180
Resolution	0	0	0	13
Mixed	0	0	0	1
Average	0.72	0.52	0.36	

Table 12: Confusion Matrix – Samsung vs. Camry

	Other	Symptom	Resolution	Mixed
Other	206	0	0	0
Solution	178	2	0	0
Resolution	13	0	0	0
Mixed	1	0	0	0

Overall, the key thing to note in the results was the difference in sample size between the different label classes. This would skew the prediction from the machine learning algorithm, giving less than desirable results.