Pranav Nakate Dr. Edward A. Fox

[npranav@vt.edu](mailto:npranav@vt.edu)

Independent Study (CS 5974)

Department of Computer Science

Blacksburg, Virginia - 24060

December 8, 2015

Dr. Edward A. Fox

Virginia Tech

B

Big Data Processing of School Shooting Archives

Contents

[List of Figures 2](#_Toc440321490)

[List of Tables 2](#_Toc440321491)

[1. Executive Summary 3](#_Toc440321492)

[2. Introduction 3](#_Toc440321493)

[3. Project Requirements 4](#_Toc440321494)

[4. Collections 6](#_Toc440321495)

[5. System Design 8](#_Toc440321496)

[5.1 Task Pipeline 8](#_Toc440321497)

[5.2 Webpage Extraction 8](#_Toc440321498)

[5.3 Webpage Cleaning 9](#_Toc440321499)

[5.4 Removing Duplicate Webpages 10](#_Toc440321500)

[5.5 Classification 10](#_Toc440321501)

[5.6 Issues and Tradeoffs 12](#_Toc440321502)

[6. User Manual 14](#_Toc440321503)

[6.1 Extracting HTML files 14](#_Toc440321504)

[6.2 Webpage Cleaning 15](#_Toc440321505)

[6.3 Removing Duplicates 16](#_Toc440321506)

[6.4 Creating Sample Sets 16](#_Toc440321507)

[6.5 Training Classifier 17](#_Toc440321508)

[6.6 Predicting Labels with Classifier 18](#_Toc440321509)

[6.7 Upload Collections to Solr 19](#_Toc440321510)

[7. Evaluation 20](#_Toc440321511)

[7.1 Alabama University Shooting 21](#_Toc440321512)

[7.2 Brazilian School Shooting 22](#_Toc440321513)

[7.3 Connecticut School Shooting 24](#_Toc440321514)

[7.4 Northern Illinois University Shooting 25](#_Toc440321515)

[7.5 Norway Shooting 27](#_Toc440321516)

[7.6 Youngstown Shooting 28](#_Toc440321517)

[7.7 Final Classification Statistics 29](#_Toc440321518)

[7.8 F1 Measure 30](#_Toc440321519)

[8. Future Work 31](#_Toc440321520)

[9. Acknowledgements 31](#_Toc440321521)

[10. References 32](#_Toc440321522)

# List of Figures

[Figure 1 Task Pipeline 8](#_Toc440202254)

[Figure 2 Cleaning Process 9](#_Toc440202255)

[Figure 3 Classification steps 11](#_Toc440202256)

[Figure 4 Alabama University Shooting 20](#_Toc440202257)

[Figure 5 Brazilian School Shooting 22](#_Toc440202258)

[Figure 6 Connecticut School Shooting 24](#_Toc440202259)

[Figure 7 Northern Illinois University Shooting 25](#_Toc440202260)

[Figure 8 Norway Shooting 26](#_Toc440202261)

[Figure 9 Youngstown Shooting 28](#_Toc440202262)

# List of Tables

[Table 1 Collections 6](#_Toc439886793)

[Table 2 Collections Statistics 7](#_Toc439886794)

[Table 3 Alabama University Shooting Results 19](#_Toc439886795)

[Table 4 Brazilian School Shooting Results 20](#_Toc439886796)

[Table 5 Connecticut School Shooting Results 22](#_Toc439886797)

[Table 6 Northern Illinois 24](#_Toc439886798)

[Table 7 Norway Shooting 25](#_Toc439886799)

[Table 8 Youngstown Shooting 26](#_Toc439886800)

[Table 9 Consolidated Statistics 27](#_Toc439886801)

[Table 10 F1 Measure Scores 28](#_Toc439886802)

# Executive Summary

The school shooting collections consist of webpages that may or may not be relevant to the events of interest. Furthermore, the webpages contain stop words such as ‘is’, ‘the’ and profane words. There are 3 main goals of this project, first is to clean the webpages, which involves getting rid of the stop words and profane words and finding the webpages relevant to the events of interest, which in the case of this project are shooting events. The third goal is to upload the cleaned and relevant webpages to Apache Solr so that they easily accessible.

The project task pipeline consists of a series of steps that are necessary to complete the 3 main tasks stated above. This project report gives details of all the steps along with their technical details. This report also describes the experiments and their results carried out during the classification stage, in which webpages are determined to be relevant or non-relevant.

# Introduction

This project is aimed at supporting the work of Sociology Professor Dr. Donald Shoemaker, co-PI on the CTRnet (Crisis, Tragedy, and Recovery network) and IDEAL (Integrated Digital Event Archiving and Library) projects. With the recent incidences of violence, school shootings in particular, it is important to understand the underlying reasons for their occurrences to prevent or at least better recover from, such incidents in the future.

The goal of the project is to take some of the data collected during the CTRnet and IDEAL projects, based at the DLRL (Digital Library Research Laboratory), such as webpage collections related to school shooting incidents, clean the collections, eliminate noise, remove non-relevant webpages, and organize them. The output webpages should be accessible for searching and browsing.

From the cleaned collection of webpages, Dr. Shoemaker should be able to answer questions like:

1. Who was the shooter?
2. What was the profile of the shooter?
3. Was the shooter captured or killed?
4. Was there a motive behind the shooting?
5. Was the shooter suffering from any psychiatric conditions?
6. If dead or killed, did the shooter leave any messages or notes?
7. What weapons were used in the shooting?
8. How did the shooter procure the weapons?
9. Who were the victims of the shooting?
10. Was there a common profile of the victims? Were they related?
11. Did the shooter know the victims?
12. What were the aftereffects of the shooting?
13. What precautions were taken to prevent such incidents in the future?

The project requirements are described in section 3. It lists the specific goals of the project. In section 4, I describe the collections that I worked with along with their statistics. I explain the details of the system design in section 5. In section 6, I have created a user manual that will help others to use the code I developed for this project. In section 5 and 6, I explain about the same processes from different perspectives. Section 5 describes the technical details of the project while section 6 describes how to use the implemented code. I describe the experiments that I carried out in the classification stage along with their results in section 7.

# Project Requirements

The goals of the project can be described as follows:

1. Identify and remove noise in the webpages
2. Process and standardize the content of the webpages
3. Remove duplicate webpages
4. Create positive and negative sample sets for each collection
5. Classify all the webpages of the collection
6. Organize the relevant webpages of the collection in Apache Solr

In the first requirement, noise can be defined as stop words and profane words. In this step, identification and removal of character sequences of stop words and profane words, which are irrelevant to the content of the page, will be carried out.

In the second requirement, the words of the text content of the webpage will be lemmatized. This will ensure that the content of all the webpages is standardized for the classification stage.

The collections have a number of duplicate webpages, which have the same title and same content. These webpages must be removed from the collection.

In the next requirement, a sample of positive webpages that are relevant to the content of the collection, and a sample of negative webpages that are not relevant, are created.

These sample sets are used to train a classifier for each collection which is subsequently used to classify all the webpages in the collection into relevant and non-relevant categories.

Finally all the relevant webpage are uploaded to Apache Solr.

# Collections

The collections are stored in WARC (Web ARChive) format. This file has an index file, under the home directory, which lists the links, types, and locations in the archive directory, of all the files in the collection. The collections have numerous types of files such as:

1. Webpages
2. Plain text files
3. PDF files
4. Image files in PNG or JPEG format
5. Video files
6. CSS files
7. Script files

I filter the links and locations of the webpages, whose type is ‘text/html’, corresponding to the first entry in this list of file types.

In this project I worked mainly on the following 6 collections:

|  |  |
| --- | --- |
| **Collection ID** | **Collection Name** |
| 1829 | Alabama University Shooting |
| 2535 | Brazilian School Shooting |
| 3437 | Connecticut School Shooting |
| 970 | Northern Illinois University Shooting |
| 2772 | Norway Shooting |
| 2379 | Youngstown Shooting |

Table Collections

The collections were created by using the Heritrix tool [5] and the web archiving service Archive-It [6] of the Internet Archive, which helps in harvesting, building, and preserving collections of digital content. The service takes URLs as input from the user. The URLs should be related to the content of the collection that is being built. For example, if we are building a collection for a particular event, then the input URLs should be of webpages that are related to the event. These URLs are used by Heritrix[5], guided by manual configuration details, and the resulting webpages are captured and stored in WARC file format. The process involved specifying several configuration parameters such as frequency of crawling, the host names that should be crawled or avoided, and the URL patterns.

Following are some statistics about these collections:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Collection** | **HTML** | **non-HTML** | **non-English** | **Duplicates** |
| **Northern Illinois University** | 73307 | 33175 | 1.04% | 43.13% |
| **Alabama University** | 30970 | 4807 | 0.25% | 37.65% |
| **Youngstown Shooting** | 11697 | 13609 | 1.80% | 38.89% |
| **Brazilian School Shooting** | 3995 | 12298 | 5.23% | 45.38% |
| **Norway Shooting** | 10321 | 36093 | -- | 36.08% |
| **Connecticut School Shooting** | 11710 | 32315 | 5.96% | 48.31% |

Table Collections Statistics

The data for non-English documents is not available for Norway Shooting, hence it is shown as a dash in the table above.

# System Design

## Task Pipeline

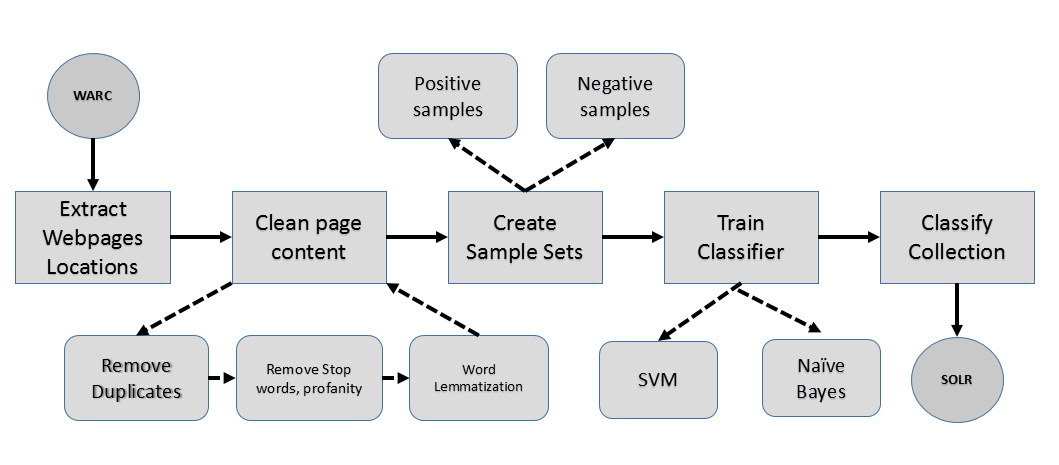


Figure Task Pipeline

The complete project pipeline is as in the figure above.

## Webpage Extraction

The collections are stored using a WARC file format. The WARC files contain numerous other types of files as listed in Section 3.

A WARC file is a compressed Zip file with all the digital content, such as HTML pages, images, CSS files, and videos stored in different folders. The compressed WARC file is expanded using the ‘unzip’ program from standard Linux utilities. For this project, I extracted only the webpages from the collection. The webpages are of type ‘text/html’ and are stored in separate files. I create a list of all the webpages in the collection along with the location of the data files in which they are stored. This list is stored in a separate file and it is used in the next process of the pipeline, which is the cleaning process.

## Webpage Cleaning

The cleaning process involves word lemmatization, stop word removal, and profanity word removal. The general flow of steps is as shown in the figure:

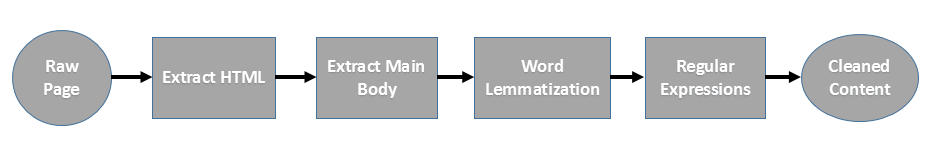


Figure Cleaning Process

In the cleaning process, using Beautiful Soup [7], we extract the main body of the page which contains the text content. Beautiful Soup parses the HTML and CSS tags of the webpage that are needed for webpage styling and navigation and we can then extract the main text content which has the information of our interest.

Stop words are commonly occurring words such as ‘is’, ‘are’, ‘the’, etc. These words do not convey any specific information but they can reduce accuracy of the classifier if the frequency of these words is high. Thus in the cleaning process we remove the stop words. The cleaned content can be viewed by people of all ages such as children. Hence it is necessary to remove the profane words from the clean content as well.

In the cleaning process, each word in the document is first lemmatized. Then the document is scanned to replace all the stop words with blank spaces by using regular expressions and then the same process is done for profanity words. For example, if we have a profane word ‘foo’ used like “foo is not a good word”, then using regular expressions the sentence would become “{profanity} is not a good word”.

The stop words and profane words are stored in separate text files. The stop words and profane words are standard lists taken from [2][3]. The cleaning script reads these files and creates a list of stop words and profane words. These lists are used along with regular expressions to remove the stop words and profane words.

## Removing Duplicate Webpages

The collections contain a lot of duplicate digital content, such as duplicate webpages, duplicate images, duplicate script files, etc. Including the duplicate content would be wasteful and hence these files must be filtered out. In the context of this project, among the webpages with exactly the same title, I selected the first webpage in the list of duplicate webpages.

## Classification

Classification is the most important aspect of this project. It involves 4 stages primarily:

1. Creating positive and negative sample sets
2. Training the classifier
3. Testing the classifier accuracy, precision, and recall
4. Running the classifier on an entire collection of webpages

To create positive and negative sample sets, the content of the cleaned samples is displayed on the screen one after the other. After I reviewed the information in the content of each webpage for about 2 minutes each page, I labelled it as positive or negative. My criteria to label a sample as positive was, if more than 30 to 40% of the information is about the shooting event, then it’s a positive sample. Otherwise I label it as a negative sample. If the page is long, and there is just a one line reference to the shooting event, then it is a negative sample. The samples are picked from the unique and cleaned collection of webpages from the previous step. I picked the labels serially from the first page of the collection.

Once the sample sets are created, the classifier is trained using these sample sets along with their labels. In this project I tested two types of classifiers, the SVM classifier and the Naïve Bayes classifier. After running a classifier, I tested the accuracy of that classifier using the same training data as unlabeled data and then checked the predicted category of these samples against their labels. If the accuracy is less than 70%, then I went back to the previous step, wherein I added more samples to the positive and negative sets and checked the accuracy again.

I repeated this process of creating sample sets for each of the collections.

The classification process first uses the CounterVectorizer function from sklearn [4] to convert the input collection of text documents to a matrix of token counts. This is shown in the figure below:

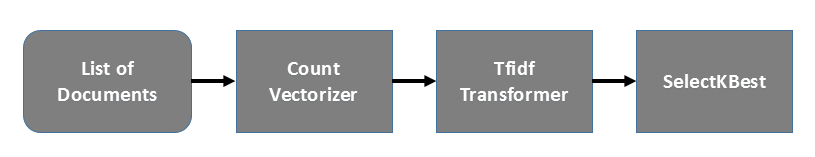


Figure Classification steps

The classification then uses a TF-IDF transformer function from sklearn [4] which transforms a count matrix to a normalized tf-idf representation. “Tf” is the term-frequency which is equivalent to word count in the case of a collection of webpages. “Idf” is the inverse document frequency that considers the number of documents in which a given word or a token appears. Tf-idf is the term frequency times inverse document frequency. This is a common weighting scheme to use in document classification as is the case here. The purpose of using tf-idf instead of simple word count of tokens in a given document is to favor words that occur often in a single document, but to scale down the impact of tokens that have high frequency in the collection and which do not add any helpful information. Further, words that occur in small numbers of webpages get more weight because of higher inverse document frequency.

The classifier then uses the SelectKBest function from sklearn [4] to select the top K features (of webpages / document) from the count matrix transformed and normalized by tf-idf transformer. This is applicable in the training phase, when the webpages from both the sample sets are used to train the classifier. It also is applicable in the prediction phase when each webpage is predicted to be relevant or non-relevant.

## Issues and Tradeoffs

1. Currently, the stop word and profanity word removal is done for all the webpages. The deduplication process is carried out after the cleaning process is completed. Cleaning duplicate webpages would be a redundant process since those pages will be discarded in the next stage. Moving the deduplication process to the start of the task pipeline would help in reducing the processing time during the cleaning process.
2. In the current deduplication process, only one webpage from among all the webpages having the same title is selected. This process does not consider the version of the webpage. It may be the case that some other page(s) with the same title have different content in the body of the page. Changes should be made so that all different pages should be kept in the collection. Deduplication should only remove webpages with identical content, and should retain the earliest webpage in a set of exact duplicates.
3. In the classification stage, while creating positive and negative sample sets, I considered samples from the beginning of the collection serially, which can introduce an improper bias. I viewed those samples and labelled them positive or negative. Thus, the samples viewed and labelled are not random in this case. Ideally the samples should be chosen randomly and then viewed and labelled.
4. The webpages may contain other information in addition to the information regarding the event of interest. For example, in a news article, one shooting event is written as a paragraph of 4 lines and the webpage has 6 other paragraphs describing other events. These paragraphs are not related to the event of interest and may not convey any useful information to a researcher who is only concerned with that particular event. If the length of this unrelated content is much larger than the main event, then the classifier may predict this as an unrelated page. Thus we might lose some webpages which can convey some useful information but are filtered out by the classifier because that content is swamped by other textual information.

# User Manual

* 1. Extracting HTML files

The collections are stored in the cluster in ZIP file format. When a collection ZIP file is expanded, the unzip program creates the collections directory. This directory is the base directory for a collection.

Run the GetFiles.py script on the collections directory. When expanded, the base folder of the collection has an \*.index.html file which lists the locations of the HTML pages stored in that collection.

These locations need to be modified with respect to the current collection folder location. The script will do that and create a new file which has the webpage link and the location where it is stored with respect to the base collections folder.

The general syntax is:

*./GetFiles.py BASE\_FOLDER INDEX\_FILE*

where,

1. BASE\_FOLDER = base folder of the collection after the WARC file is unzipped
2. INDEX\_FILE = output index file name for the collection

For example,

*./GetFiles.py 2772 InputFiles.txt*

Here 2772 is the base folder of the collection with ID 2772. The IDs of all the collections are listed in Table 1 Collections. Now we have the locations of all the HTML files stored in the collection in InputFiles.txt.

* 1. Webpage Cleaning

The next step is to clean all these webpages, to remove noise that includes stop words, profane words, HTML tags, CSS and JavaScript tags, etc. To do this, run the webpageclean\_new.py which will take the index file created in the step described in Section 5.1 and clean either the full collection, or, if such is necessary, a specified number of webpages.

The general syntax is:

*./webpageclean\_new.py INDEX\_FILE OUTPUT\_FILE FILE\_COUNT*

where,

1. INDEX\_FILE = index file of the collection created in step 1
2. OUTPUT\_FILE = the output file containing all the cleaned pages of the collection
3. FILE\_COUNT = a parameter to specify the number of webpages to clean from the beginning of the collection, without cleaning the entire collection

For example, to clean the collection with ID 2772 using the index file created in the above step, execute the command:

*./webpageclean\_new.py InputFiles.txt 2772\_clean\_complete.txt -1*

The first argument is the index file created in the prior step, second is the output file where all the cleaned webpages will be stored in separate lines, and third is a file count argument. If a file count is specified, then it will only clean file count number of webpages. I created this functionality, in case I want to quickly clean up a few pages from the index file. If file\_count is negative, as in the given example, then it will clean the entire collection. The above command will clean all the HTML files listed in the index file [InputFiles.txt] of collection 2772.

The output file is one consolidated and concatenated file containing all the pages in the collection. One page is represented by 2 lines each of this file. The first of these 2 lines, contains the header information with tab separated fields. This line is of the format:

*--webpage-- PAGE\_TITLE PAGE\_DATE PAGE\_LINK*

The second line contains the entire content of the page.

* 1. Removing Duplicates

To remove duplicates from the collection, I wrote a script named Dedup.py. This script iterates through all the webpages in the list, and if a webpage with its title is not yet present in the unique pages collection then it adds that page. All subsequent pages with the same title as this page’s title are discarded.

To run the deduplication script, the syntax is:

./Dedup.py COLLECTION\_FILE UNIQUE\_PAGES\_FILE

where,

1. COLLECTION\_FILE = the output file of the prior step, containing the cleaned webpages
2. UNIQUE\_PAGES\_FILE = the output file of this stage; it will contain all the cleaned unique webpages of the collection

For example, to remove duplicates from the cleaned collection file of the step in section 5.2, use:

./Dedup.py 2772\_clean\_complete.txt 2772\_unique.txt

* 1. Creating Sample Sets

To create training sets, I have created a CreateSamples.py script. This script will take the file in which all unique and cleaned HTML pages are stored in the step of section 5.3. It will create a set of positive samples and a set of negative samples. It will show a HTML page's content on the screen, and based on the input given by the user for that content, it will put that HTML page in either positive or negative training sample set. The command is:

./CreateSamples.py UNIQUE\_PAGES\_FILE RELSampleFILE NonRELSampleFILE OFFSET

where,

1. UNIQUE\_PAGES\_FILE = the file containing cleaned unique pages of the collection in the prior step
2. RELSampleFILE = the output file containing the positive samples
3. NonRELSampleFILE = the output file containing the negative samples

An example is:

./CreateSamples.py 2772\_clean\_complete.txt sample+.txt sample-.txt 0

The offset, if non-zero, will skip the first offset number of pages, and start from the offset + 1 page. I created this, in case I create sample sets from same output file at different times. I can just specify an offset and start from where I left off, so that I don't have to look at the previous pages again.

* 1. Training Classifier

Once we have positive and negative training data, we classify the complete cleaned collection. To do this, use:

./ClassifyNew.py RELSampleFILE NonRELSampleFILE UNIQUE\_PAGES\_FILE FLAG REL\_FILE NonREL\_FILE K

where,

1. RELSampleFILE = the positive sample file created in the prior step
2. NonRELSampleFILE = the negative sample file created in the prior step
3. UNIQUE\_PAGES\_FILE = the cleaned unique pages of the collection created in the step described in Section 5.3
4. FLAG: If the classifier is being only trained, then this flag will be False; if the classifier is being trained and also used to predict labels of all the pages in the collection then this flag will be True
5. REL\_FILE = the output file containing the webpages being classified as relevant. In the case of classifier training, this file will be empty.
6. NonREL\_FILE = the output file containing the webpages being classified as non-relevant. In the case of classifier training, this file will be empty.
7. K = value of the k for the KBest feature selection function

For example, for training the classifier, use:

./ClassifyNew.py sample+\_new.txt sample-\_new.txt 2772\_clean\_complete.txt **False** REL.txt NON\_Rel.txt 20

As highlighted in bold above, the flag is false for just training the classifier and checking what accuracy it gives with the current size of positive and negative sample sets.

* 1. Predicting Labels with Classifier

Once we get good accuracy, the same command as above can be used to classify the whole collection with flag equal to True.

For example, use:

./ClassifyNew.py sample+\_new.txt sample-\_new.txt 2772\_clean\_complete.txt **True** REL.txt NON\_Rel.txt 20

Note that the flag in the command above is set to True, so that the classifier will be trained using the sample set files AND then it will also predict labels for all the unique pages in the collection. The relevant pages will be stored in REL.txt and non-relevant pages will be stored in NON\_Rel.txt. The pages will be stored in these files in the 2 line format as described in section 5.2.

* 1. Upload Collections to Solr

This is the final step in the pipeline. I have created a script SolrInterface.py that will take the concatenated file of relevant webpages created in step of section 5.6 and will upload each page to the Apache Solr. The syntax to use the script is:

./SolrInterface.py REL\_FILE EVENT\_NAME EVENT\_ID

Where,

REL\_FILE is the concatenated file of all the relevant webpages as classified by the trained classifier in the step of section 5.6,

EVENT\_NAME is the name of the shooting event

And EVENT\_ID is the id of the shooting collection as listed in Table 1 Collections.

For example,

./SolrInterface.py 970\_REL.txt Northern\ Illinois\ University\ Shooting 970

This command above will upload the relevant pages from collection with id 970 and event name “Northern Illinois University Shooting” to Apache Solr. The pages are stored in 2 line format in the concatenated file 970\_REL.txt.

# Evaluation

For evaluation, I ran experiments for each collection separately. In each experiment I used different values of K, which is an input parameter for the SelectKBest function. SelectKBest is used to select the top K best features of a webpage, which are used in the process of training and classification of the classifier. The features of the webpage are the words and the values of these features is the count of these words in the webpage. This is explained in detail in section 4.5. By changing values of K, one can change how many top features or high frequency words from a webpage are used in the training and classification of the classifier.

Using different values of K, 2 classifiers, Support Vector Machines (SVM) and Naïve Bayes (NB), are trained. Their accuracy is tested on the training data and test data. The training data is created by reading 75% of the positive samples and 75% of the negative samples. The remaining 25% of both sample sets is test data. To train the classifier training samples along with their respective labels are fed to the classifier. Then the same training data without labels is used to test the accuracy of the classifier. The accuracy is also calculated by using the classifier to predict labels of samples in test data and then verifying them against their actual labels.

In the following subsection results of experiments to find the accuracy of the classifier using different values of K are given for each collection. Accuracy is the simplest measure to test the reliability of the classifier. Given the labelled training data, accuracy measure would indicate how many of the labelled samples get correctly classified in the same category as their labels. Precision and Recall, is another measure to find the reliability of the classifier.

* 1. Alabama University Shooting

Following are the results of running experiments for accuracy on this collection.

Size of positive sample set = 81

Size of negative sample set = 145

For this collection, I initially tested the classifier with equal size of positive and negative sample sets. But the accuracy in that case was lower than 60%. When I was viewing the samples, I observed that the relevant pages were longer while the non-relevant were short. Hence by adding more negative samples I was able to increase the accuracy of the classifier above 70% as can be seen in the table below. As mentioned in section 4.5, K denotes the number of top K best features or in this case the top K best words of the webpage. For running the experiments, I used the range of K up to 100 thereby choosing at most top 100 features or words from the webpage.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **K** | **Training (SVM)** | **Test (SVM)** | **Training (NB)** | **Test (NB)** |
| 5 | 0.9059 | 0.9107 | 0.6529 | 0.6428 |
| 10 | 0.9647 | 0.9285 | 0.7705 | 0.6964 |
| 15 | 0.9764 | 0.9642 | 0.8235 | 0.75 |
| 20 | 0.97058 | 0.9642 | 0.8705 | 0.78571 |
| 25 | 0.9058 | 0.8928 | 0.8883 | 0.8035 |
| 50 | 0.6411 | 0.6428 | 0.9588 | 0.9464 |
| 75 | 0.6411 | 0.6428 | 0.9470 | 0.9642 |
| 100 | 0.6411 | 0.6428 | 0.9470 | 0.9642 |

Table Alabama University Shooting Results

**Results of experiments to find accuracy of the classifier using different values of K**

Figure Alabama University Shooting

The final classification is done using an SVM classifier with K = 15 using the following command in the 1829\_new folder:

../ClassifyNew.py 1829\_positive.txt 1829\_negative.txt 1829\_ALL.txt True 1829\_REL.txt 1829\_NON\_REL.txt 15

The relevant samples will be in file 1829\_REL.txt after the command executes.

* 1. Brazilian School Shooting

Following are the results of running accuracy experiments on this collection:

Size of positive sample set = 60

Size of negative sample set = 160

For this collection, when I was reviewing the samples to create training sets, I observed that the relevant pages were long webpages from news sites while the non-relevant pages were short and mostly consisted of blogs and discussion forum content. Hence the accuracy of the classifier was high for lower values of K. Initially I tested the classifier with 100 samples each in positive and negative set, but I could not get the accuracy above 70% in that case. Therefore I tried by changing the sizes of the positive and negative sample sets, and with the 60 positive samples and 160 negative sample, I could get accuracy greater than 70%, hence I chose to keep these sizes of the training sets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **K** | **Training (SVM)** | **Test (SVM)** | **Training (NB)** | **Test (NB)** |
| 5 | 0.9636 | 0.9818 | 0.7272 | 0.7272 |
| 10 | 0.9575 | 0.9272 | 0.7272 | 0.7272 |
| 15 | 0.9090 | 0.8363 | 0.7272 | 0.7272 |
| 20 | 0.7818 | 0.7818 | 0.7272 | 0.7272 |
| 25 | 0.7333 | 0.7272 | 0.7272 | 0.7272 |
| 50 | 0.7272 | 0.7272 | 0.7272 | 0.7272 |
| 75 | 0.7272 | 0.7272 | 0.7272 | 0.7272 |
| 100 | 0.7272 | 0.7272 | 0.7272 | 0.7272 |

Table Brazilian School Shooting Results

**Results of experiments to find accuracy of the classifier using different values of K**

Figure Brazilian School Shooting

The final classification is done using an SVM classifier with value of K = 5 using the following command in the 2535\_new folder:

../ClassifyNew.py 2535\_p.txt 2535\_n.txt 2535\_ALL.txt True 1829\_REL.txt 1829\_non\_REL.txt 5

The relevant pages will be in 1829\_REL.txt.

* 1. Connecticut School Shooting

Following are the results of running accuracy experiments for this collection.

Size of positive sample set = 200

Size of negative sample set = 200

In this collection, I initially tested the classifier for accuracy with 100 positive and 100 negative samples. The accuracy in that case was lesser than 70% for all values of K. The accuracy increased above 80% for all values of K when I doubled the size of positive and negative training sets. Hence I chose to go forward with 200 positive and 200 negative samples for training the classifier.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **K** | **Training (SVM)** | **Test (SVM)** | **Training (NB)** | **Test (NB)** |
| 5 | 0.8866 | 0.92 | 0.7366 | 0.81 |
| 10 | 0.8733 | 0.92 | 0.9633 | 0.92 |
| 15 | 0.8533 | 0.91 | 0.94 | 0.92 |
| 20 | 0.8733 | 0.92 | 0.9433 | 0.92 |
| 25 | 0.89 | 0.93 | 0.94 | 0.92 |
| 50 | 0.96 | 0.96 | 0.9166 | 0.9 |
| 75 | 0.9566 | 0.96 | 0.8766 | 0.88 |
| 100 | 0.96 | 0.96 | 0.8833 | 0.87 |

Table Connecticut School Shooting Results

**Results of experiments to find accuracy of the classifier using different values of K**

Figure Connecticut School Shooting

The final classification is done using an SVM classifier with value of K = 15 using the following command in the 3437\_new folder:

../ClassifyNew.py 3437\_p.txt 3437\_n.txt all.txt True 3437\_REL.txt 3437\_non\_REL.txt 15

The relevant pages are listed in 3437\_REL.txt

* 1. Northern Illinois University Shooting

Following are the results of accuracy experiments done for this collection.

Size of positive sample set = 166

Size of negative sample set = 160

For this collection, while reviewing the samples to create positive and negative sample sets, I observed that the relevant and non-relevant pages were of same length from diverse sources such as news sites, local websites. I initially had 100 samples each in the positive and negative sets, but accuracy was lesser than 75%. I kept on increasing the samples in both sets, and I was able to get above 75%, with 166 positive and 160 negative samples in the training sets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **K** | **Training (SVM)** | **Test (SVM)** | **Training (NB)** | **Test (NB)** |
| 5 | 0.8979 | 0.9382 | 0.6244 | 0.6049 |
| 10 | 0.951 | 0.9753 | 0.7387 | 0.7407 |
| 15 | 0.9551 | 0.9753 | 0.7428 | 0.7407 |
| 20 | 1 | 1 | 0.7428 | 0.7283 |
| 25 | 0.7591 | 0.7283 | 0.7428 | 0.7283 |
| 50 | 0.6040 | 0.5925 | 0.8693 | 0.8518 |
| 75 | 0.5102 | 0.5061 | 0.8775 | 0.8888 |
| 100 | 0.5102 | 0.5061 | 0.8775 | 0.8888 |

Table Northern Illinois

**Results of experiments to find accuracy of the classifier using different values of K**

Figure Northern Illinois University Shooting

The final classification is done using an SVM classifier with K = 20 and using the following command in the 970\_new folder:

../ClassifyNew.py 970\_p.txt 970\_neg.txt ALL\_NEW.txt True 970\_REL.tx 970\_non\_REL.txt 20

The relevant pages are listed in 970\_REL.txt after the command executes.

* 1. Norway Shooting

Following are the results for accuracy experiments for this collection.

Size of positive sample set = 227

Size of negative sample set = 230

For this collection, when I was reviewing the samples to create positive and negative sample sets, I observed that relevant pages and non-relevant pages were of equal lengths and were mostly from news sites. Lot of the non-relevant pages were from discussion forums or broken pages with incomplete and non-coherent content. I initially had 150 samples in both positive and negative sample sets, but the accuracy of the classifier was lesser than 70%. I kept on increasing the sample set sizes, and I could get more than 75% accuracy when I had roughly 230 samples in both positive and negative sample sets.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **K** | **Training (SVM)** | **Test (SVM)** | **Training (NB)** | **Test (NB)** |
| 5 | 0.7819 | 0.6991 | 0.5610 | 0.5132 |
| 10 | 0.7761 | 0.6991 | 0.7558 | 0.6637 |
| 15 | 0.7703 | 0.6902 | 0.7441 | 0.6548 |
| 20 | 0.7587 | 0.6902 | 0.7645 | 0.6725 |
| 25 | 0.7470 | 0.6814 | 0.8895 | 0.7787 |
| 50 | 0.7383 | 0.6814 | 0.8720 | 0.8407 |
| 75 | 0.6133 | 0.5575 | 0.8488 | 0.8407 |
| 100 | 0.5319 | 0.5309 | 0.8488 | 0.8495 |

Table Norway Shooting

**Results of experiments to find accuracy of the classifier using different values of K**

Figure Norway Shooting

The final classification is done using an SVM classifier with K = 15 by using the following command in the 2772\_new folder:

../ClassifyNew.py sample+\_new.txt sample-\_new.txt 2772\_all.txt False 277\_REL.txt 2772\_non\_REL.txt 15

The relevant files are in 2772\_REL.txt after the command executes.

* 1. Youngstown Shooting

Following are the results of running experiments to find the accuracy of the classifier using different values of K.

Size of positive sample set = 115

Size of negative sample set = 110

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **K** | **Training (SVM)** | **Test (SVM)** | **Training (NB)** | **Test (NB)** |
| 5 | 0.7294 | 0.6 | 0.5117 | 0.5090 |
| 10 | 0.9411 | 0.8545 | 0.5117 | 0.5091 |
| 15 | 0.8882 | 0.8727 | 0.5117 | 0.5090 |
| 20 | 0.5117 | 0.5090 | 0.5 | 0.5090 |
| 25 | 0.5117 | 0.5090 | 0.5058 | 0.5091 |
| 50 | 0.5117 | 0.5090 | 0.5764 | 0.5272 |
| 75 | 0.5117 | 0.5090 | 0.5941 | 0.5818 |
| 100 | 0.5117 | 0.5090 | 0.6529 | 0.6181 |

Table Youngstown Shooting

**Results of experiments to find accuracy of the classifier using different values of K**

Figure Youngstown Shooting

The final classification is done using an SVM classifier using K = 15 with the following command in the 2379\_new folder:

../ClassifyNew.py 2379\_p.txt 2379\_n.txt all.txt True 2379\_REL.txt 2379\_non\_REL.txt 15

The relevant webpages will be in 2379\_REL.txt after the command completes.

* 1. Final Classification Statistics

Following are the statistics of the collections after final classification. The final classification implies using the trained classifier on the entire classification to predict their labels, which are relevant or non-relevant page.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Collection ID** | **Collection Name** | **Relevant** | **Non-relevant** | **Total Pages** |
| 1829 | Alabama University Shooting | 1.406 | 98.593 | 6470 |
| 2535 | Brazilian School Shooting | 8.839 | 91.160 | 1120 |
| 3437 | Connecticut School Shooting | 17.541 | 82.458 | 3238 |
| 970 | Northern Illinois University Shooting | 26.733 | 73.266 | 15385 |
| 2772 | Norway Shooting | 13.451 | 86.543 | 7419 |
| 2379 | Youngstown Shootings | 40.03 | 59.964 | 3427 |

Table Consolidated Statistics

* 1. F1 Measure

Following are the statistics of F1-measure. The value of K used to measure these scores is 15, since for most collections this is the value of K used to train the classifier and then used it to predict if a page is a relevant or non-relevant page from the collection.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Collection** | **Precision** | **Recall** | **F1-score** | **Support** |
| Northern Illinois University | 0.83 | 0.74 | 0.72 | 81 |
| Alabama University | 0.82 | 0.75 | 0.7 | 56 |
| Youngstown Shooting | 0.26 | 0.51 | 0.34 | 55 |
| Brazilian School Shooting | 0.53 | 0.73 | 0.61 | 55 |
| Norway Shooting | 0.73 | 0.65 | 0.62 | 113 |
| Connecticut School Shooting | 0.83 | 0.74 | 0.72 | 81 |

Table F1 Measure Scores

# Future Work

1. In the webpage classification process, the title of the webpage can be used as one of the features for classification.
2. The webpage can be broken down into paragraphs, and the classifier can be applied to the paragraphs instead, or in addition to the whole document, to extract all the relevant information about the event.
3. One of the features of the classifier can be the length of the webpage in terms of the number of words in the text content of the webpage. Further, short webpages can be classified in one way, and larger webpages can be classified separately, if it turns out that such would be more effective.
4. Apart from the webpages, other types of files in the collection, such as text and PDF files, could be processed to extract additional useful information related to the event of interest.
5. Since we have a multi-node cluster available in DLRL, the classification can be done in parallel on different pages of the collection. For this the classification script can be modified to be written using MapReduce paradigm. For really large collections, this approach will speed up the process of doing the classification on the entire collection.

# Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. NSF - IIS1319578: Small: Integrated Digital Event Archiving and Library (IDEAL).

This material is based upon work supported by the National Science Foundation under Grant No. NSF - IIS1319578: Small: Integrated Digital Library Support for Crisis, Tragedy, and Recovery.

I would like to thank Dr. Fox for giving me this wonderful opportunity. I would also like to thank the entire DLRL group. Special thanks go to Mohamed, Sunshin, and Prashant for all their help in the project. I hope this project helps Dr. Shoemaker in his research.

# References

1. Reducing Noise for IDEAL, Prashant Chandrasekar, Wang Xiangwen, <http://vtechworks.lib.vt.edu/handle/10919/52340>
2. Stanford stop word list, <https://github.com/stanfordnlp/CoreNLP/blob/master/data/edu/stanford/nlp/patterns/surface/stopwords.txt> accessed on 10/05/2015
3. Google list of bad words, <http://fffff.at/googles--official--list--of--bad--words/> accessed on 10/05/2015
4. Sklearn, <http://scikit-learn.org/stable/> accessed on 10/12/2015
5. Heritrix, <http://www.crawler.archive.org/index.html> accessed on 12/26/2015
6. ARCHIVE-IT, <https://archive-it.org/> accessed on 12/26/2015
7. Beautiful Soup, [https://pypi.python.org/pypi/beautifulsoup4/4.3.2 accessed on 12/27/2015](https://pypi.python.org/pypi/beautifulsoup4/4.3.2%20accessed%20on%2012/27/2015)