CSC 526 Project Report

Web Crawler Malware Detection System

Austin Sanders

aws1222@jagmail.southalabama.edu

Introduction

The purpose of this project is to develop a classification model that determines whether a website is either malicious or benign. Detecting malicious websites can be advantageous in cyber security as the web "has been one of the most effective mechanism for criminals to distribute malicious code" [1]. Today's web possess many challenges in the dependency of the data being analyzed [2]. As webpages become more dynamic in state changes, current detection algorithms may overlook potentially malicious websites. This project may offer novel solutions regarding dynamically evolving websites by analyzing elements that may elicit malicious state changes (such as DOM-modifying methods/functions), allowing malware to infect the host. The following are a few common challenges with current malicious websites [2]: state explosion, state navigation, triggering state changes, and unreachable states.

Particularly, this project explores the similarities of tokenized website HTML document characters following the classical term frequency inverse document frequency (TF-IDF) algorithm, using a text vector space. The TF-IDF algorithm is common method as an information retrieval and extraction subtask used to support content analysis of web pages for text categorization [3]–[5]. Furthermore, the TF-IDF algorithm takes a 'bag of words' approach [4]. The 'bag of words' is lazy learning method that does not assume any sequential relationship among tokenized elements. The 'bag of words' is often used with k-nearest neighbors, as text categorization is generally a supervised classification method where known labels of text categorized are trained to predict unknown labels of newly retrieved documents [6]. Moreover, TF-IDF is a statistical technique that identifies strong words from a corpus and gives it a larger weight than words that are common [3]. In turn, this aids distance measures in distinguishing dissimilarities in text categorizations.

Web mining is the process of discovering useful information from the World Wide Web [7]. According to [4], web mining has cross between multiple fields in the knowledge discovery process, such as database, information retrieval, and machine learning. Search engines, such as Google, use web mining techniques to provide relevant websites given a search query. Search engines are rooted in the crawlers that collect information from websites that are indexed for search queries. The purpose of the web crawler is to cover as much website content as possible while bypassing irrelevant websites to the search query [8]. However, there are no standards for developing websites [8] which is problematic for web crawlers and can also result in hidden threats. This considered, the application of the TF-IDF algorithm for this project vectorizes parsed character elements to find strong discriminant characters to identify and index a website as malicious or benign. The extension of the TF-IDF algorithm introduced with this project may be used with automated browsing services, such as Selenium, to alert users of probable threat. This could help with identifying and filtering incoming traffic to a network by triggering probable threats from malicious domains.

The framework for this project is rooted in the knowledge discovery diagram (KDD) and a modification of the framework presented in [9]. Figure 1 is an illustration of the KDD diagram. The KDD diagram contains a series of step undertaken to provide meaningful information from 'nuggets' of data [10]. The key components of the KDD diagram typically include data cleaning, data integration, data selection, data transformation, data mining, pattern evaluation, and visualization.

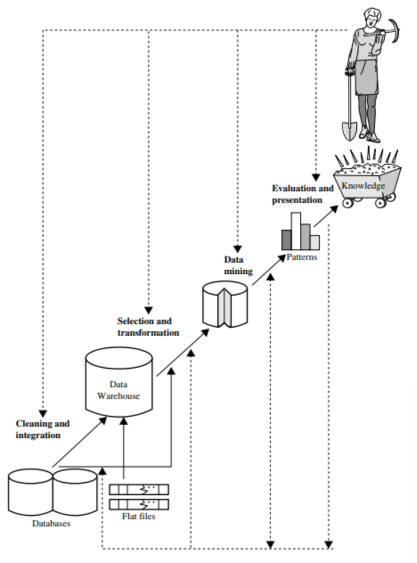


Figure 1. Knowledge Discovery Diagram [10]

Proposed Solution

Following the classical TF-IDF algorithm, this project collects and parses h1 tags from HTML files, vectorizes parsed h1 tag characters, and calculates the similarity between vectors using the Euclidean and Manhattan distance to find natural groups which are classified as malicious or benign with the KNN algorithm. The classification criteria for the KNN algorithm is determined by the majority vote of the class label within the group cluster of the trained data. The unknown class label of a test instance is given a class label having the majority vote from the trained model with the group where the instance belongs. The figure 2 below is a summary of the framework used for this project:

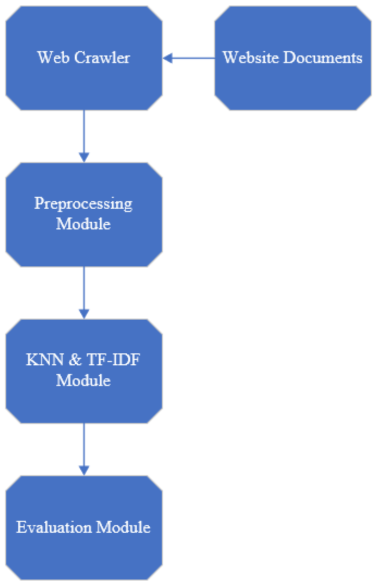
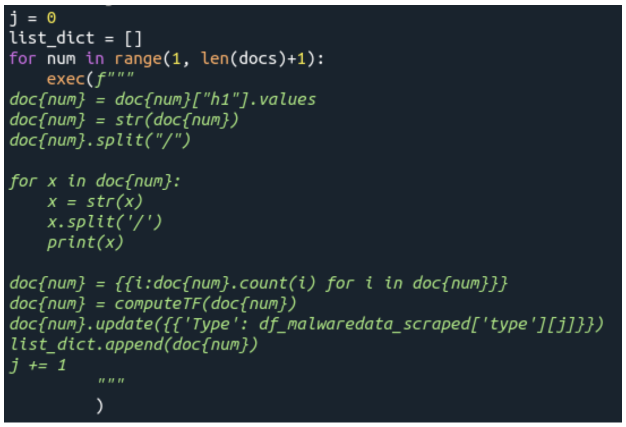


Figure 2. Project Framework

First, a web crawler is used to gather the necessary information from website documents for the experiment. The text gathered for each website is cleaned by removing any null values for h1 tags and preprocessed using string functions where every element is split and vectorized as a python dictionary having key value pairs (character: count). Note that the cleaning step of the KNN diagram is grouped as preprocessing with the framework presented in figure 2. Afterwards, the Python dictionaries values are transformed using the TF-IDF algorithm. This is a key step, as the text representation is an important part of preprocessing [3] since the represented text helps with discrimination power. Once the data is preprocess and transformed, the KNN algorithm is used to discover and classify the similarity of the document vectors. Figure 3 below is a snipping python code used to vectorize the character elements of h1 tags.



```
j = 0
list_dict = []
for num in range(1, len(docs)+1):
    exec(f"""
doc{num} = doc{num}["h1"].values
doc{num} = str(doc{num})
doc{num}.split("/")

for x in doc{num}:
    x = str(x)
    x.split('/')
    print(x)

doc{num} = {{i:doc{num}.count(i) for i in doc{num}}}
doc{num} = computeTF(doc{num})
doc{num}.update({{'Type': df_malwaredata_scraped['type'][j]}})
list_dict.append(doc{num})
j += 1
         """
         )
```

Figure 3. Website h1 Tag Character Vectorization

Figure 4 illustrates an Andrews Curve that helps visualize the differences in the two web documents categories (malicious (1) vs benign (0)) analyzed for this project. The Andrews Curve is plotted after applying the TF-IDF algorithm. Each spike in the visual represents the respective character (feature) value of the document vectors. Figure 5 demonstrates that the benign websites analyzed for this project have substantially higher character values compared to malicious websites. However, the small amounts of red suggest that there are character values with discriminant power.

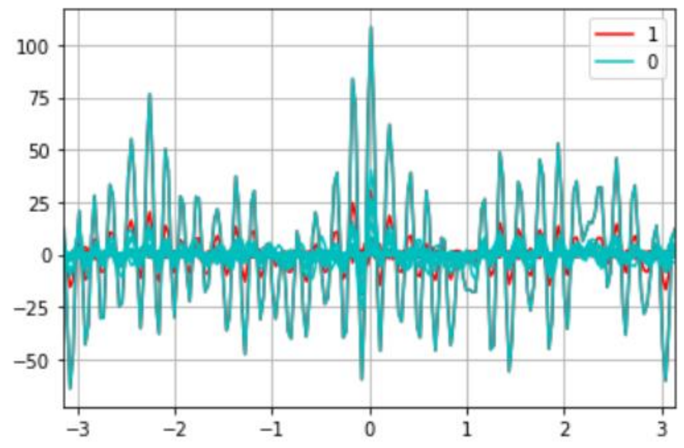
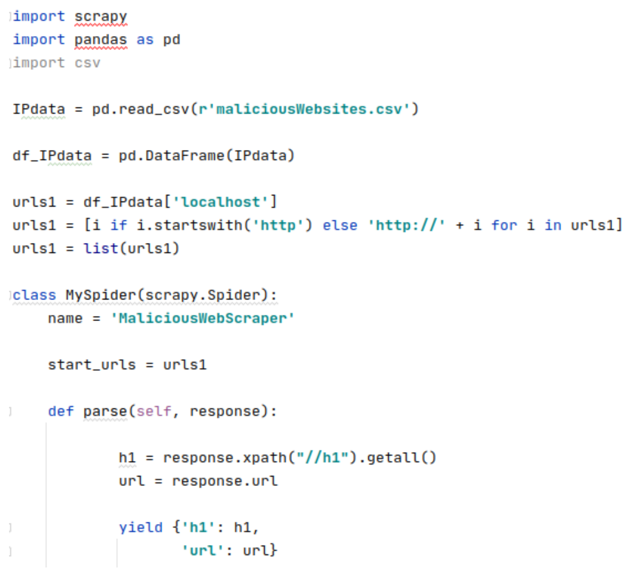


Figure 4. Document Character Vector Visualization Malicious vs Benign

The TF-IDF algorithm typically is used to tokenize words for text classification for semantic analysis and natural language processing [11]; however, this project extends the TF-IDF to analyze character values for threat analysis. Referenced links, for example, may be the source of malware and may be triggered by a state change. This noted, analyzing individual characters may be more useful in terms of threat analysis instead of content analysis. Analyzing contents of the semantic web may be useful for finding and indexing relevant content for users which involves the text of the website rather than character values. Some characters involved in the makeup of the source code of the website may not be visible to the user, but the character elements that can possibly trigger malicious state changes is likely more relevant for current, dynamic threat detection. For this reason, the TF-IDF algorithm was modified to add weight to discriminant characters, rather than words (see source code).

Experimental setup

Scrapy is a Python web scraping library used to develop user defined web crawlers (called spiders with the scrapy documentation [12]) that request and parse website downloads that can be analyzed both statically and dynamically. The user defined crawlers (spiders) retrieve h1 tags from about 500 malicious IP address provided from a blacklist (malewaredomainlist.com Host List – Active) as well as the top 500 benign websites. Available list of recent and active IP addresses that have been identified as being malicious and the benign websites are scraped through a Linux virtual machine provided by Oracle Virtual Box. Venv is a Python library which also provides a virtual environment that is used to store program files as an extra layer of security. Figure 5 below is the user-defined crawler created to scrape the malicious websites domains provided from malewarehostlist.com. A separate spider is created for the benign domains.



```
import scrapy
import pandas as pd
import csv

IPdata = pd.read_csv(r'maliciousWebsites.csv')

df_IPdata = pd.DataFrame(IPdata)

urls1 = df_IPdata['localhost']
urls1 = [i if i.startswith('http') else 'http://' + i for i in urls1]
urls1 = list(urls1)

class MySpider(scrapy.Spider):
    name = 'MaliciousWebScraper'

    start_urls = urls1

    def parse(self, response):

            h1 = response.xpath("//h1").getall()
            url = response.url

            yield {'h1': h1,
                   'url': url}
```

Figure 5. Malicious Web Crawler Used for this Project

For a better understanding of the Scrapy framework, figure 6 provides a visual depiction of the framework. This image can be found in the documentations at scrapy.org. This project focuses primarily on the user defined spiders (1) which requests currently only h1 tags from downloaded documents from the Internet. Each spider has a user-agent that requests documents from a domain server with a proxy address. For ethical purposes, the robots.txt file is set to obey to avoid any potential legal conflicts. The robots.txt configuration may be changed to false, if desired. There are other benefits to this framework, such as middleware and data pipelining and will be discussed in the limitation and further work section.

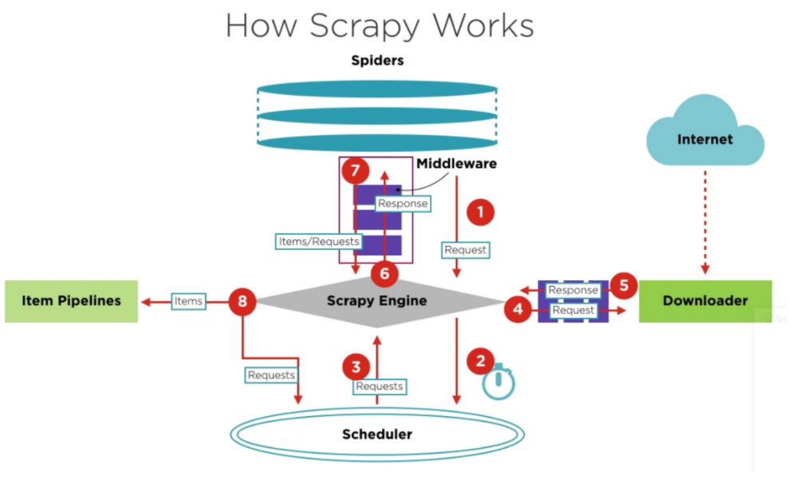
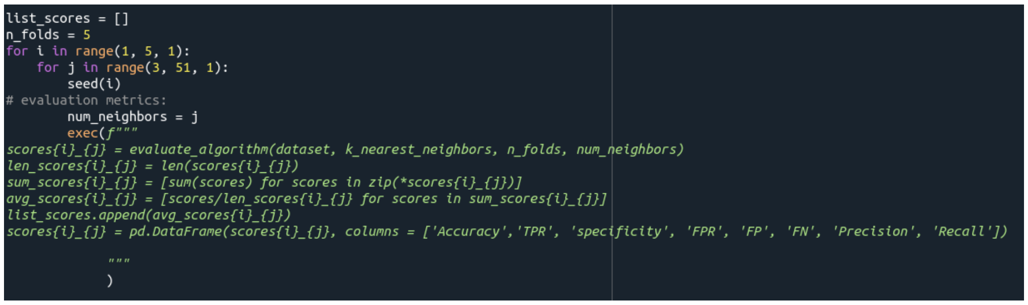


Figure 6. Scrapy Framework [12]

The evaluation metrics for determining success and failure are the following: accuracy, true positive rate, specificity, false positive rate, false negatives, precision, and recall. Additionally, cross-validation is used where the results of the evaluations are the aggregate means. The evaluation metrics are executed through a nested for-loop with increments in the seed and number of neighbors to tune the parameters for better results. Figure 7 is a snipping of the python code used for the evaluation metrics.



```
list_scores = []
n_folds = 5
for i in range(1, 5, 1):
    for j in range(3, 51, 1):
        seed(i)
# evaluation metrics:
        num_neighbors = j
        exec(f"""
scores{i}_{j} = evaluate_algorithm(dataset, k_nearest_neighbors, n_folds, num_neighbors)
len_scores{i}_{j} = len(scores{i}_{j})
sum_scores{i}_{j} = [sum(scores) for scores in zip(*scores{i}_{j})]
avg_scores{i}_{j} = [scores/len_scores{i}_{j} for scores in sum_scores{i}_{j}]
list_scores.append(avg_scores{i}_{j})
scores{i}_{j} = pd.DataFrame(scores{i}_{j}, columns = ['Accuracy','TPR', 'specificity', 'FPR', 'FP', 'FN', 'Precision', 'Recall'])

             """
             )
```

Figure 7. Nested For-Loop to Tune KNN Parameters

Results

Overall, the results of the evaluation metrics reveal that this project provides a sound baseline to further build and scale up. According to the receiver operating characteristics (figure 9 and figure 11), all the parameters given in the nest for-loop give a better than random predictions since all the plotted results are to the left of the diagonal red line. The farther the plotted points are to the top left corner indicates a better classification model. Typically, the plotted points on a ROC curve is based on changes in threshold values for the decision criteria [13]. The changes in the decision criteria, such as changes in the sigmoid function with logic regression model, can produce a curve. In the case of this project, the ROC is a derivation of this concept. Instead of changing the decision criteria, changes in the number of neighbors and the seed are used to visualize difference in performance. A scatter plot, instead of a curve, looks more appealing. Additionally, difference in the Euclidean distance and the Manhattan distance functions are compared. The visuals suggest that there are minimal changes between the uses of these two distance function. By identifying the points with the best performance, the classification model can be tuned to those parameters.

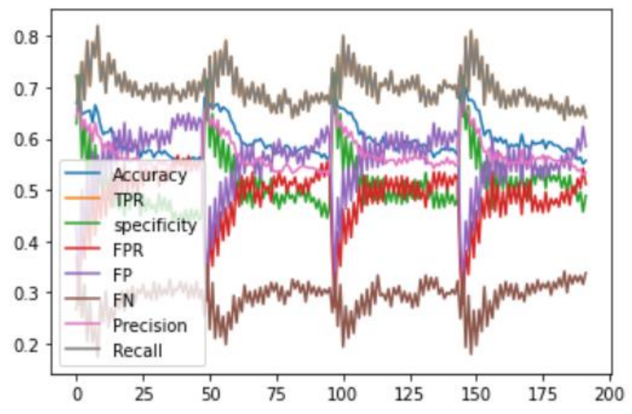


Figure 8. Evaluation Metrics Euclidean Distance

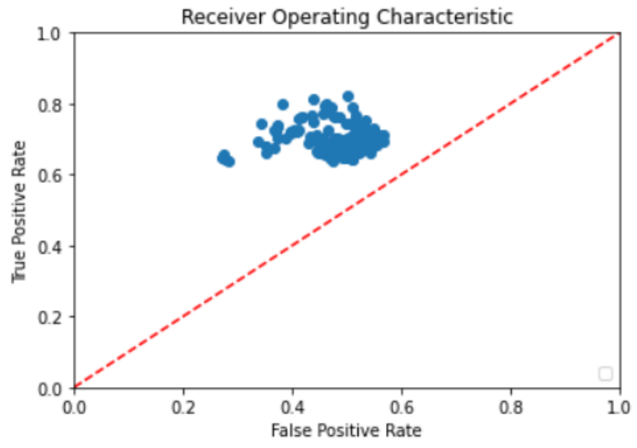


Figure 9. ROC Scatter Plot Euclidean Distance

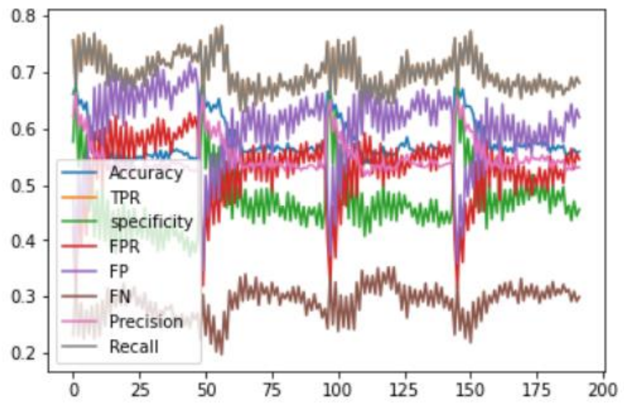


Figure 10. Evaluation Metrics Manhattan Distance

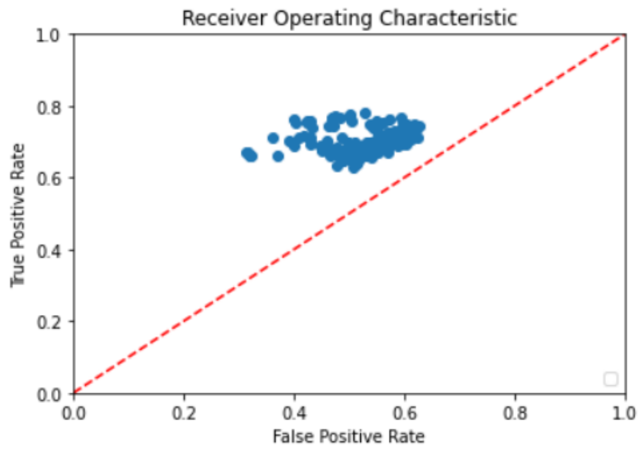


Figure 11. ROC Scatter Plot Manhattan Distance

Limitation and proposed solution:

Considering the time constraint for the completion of this project, it is limited in scope. As such, this project provides a baseline as an exploratory study where future research can build towards a scalable system. The limitations for this study are primarily regarding scalability and memory leaks which are challenges from websites pending download from crawler requests, too many requested information, and high dimensional document character vectors. The Scrapy framework provides possible solutions through middleware and timers. Websites that take too long to download are returned as “None”. Returning a “None” can at least be used to further analyze websites regarding this issue [12].

Furthermore, Scrapy provides solutions for data pipelining. Piping data to a database was not considered for this project, but it can be used for expansion with more data from source codes. The data analyzed for this study was only regarding h1 tags, but other elements of an HTML file can be gathered and stored as entities in a relational database. A possible expansion with more data collected from different HTML tags can be relating to the mapping of ontologies associated with different entities, such as presented in [14]. The mapping of ontologies can potentially be further established between different categories of malicious websites (class labels).

The TF-IDF algorithm also has its limitations. While this algorithm can distinguish strong discriminant characters against the corpus, it does not distinguish the character power within document categories [3], such as within benign or malicious websites. So, for future work, the discriminant characters values can be further refined to text categories as illustrated in [3]. Moreover, the vectorizing characters can create a high dimensional vector space resulting in computational cost. As such, dimension reduction techniques should be considered. One possible approach can be with a text manifold, as described in [15]. Figure 12 is an illustration of this concept. Lastly, with big data, parallel processing may help performance. [5] proposes using MapReduction with Hadoop for parallel processing which could lead to faster convergence and scalability.

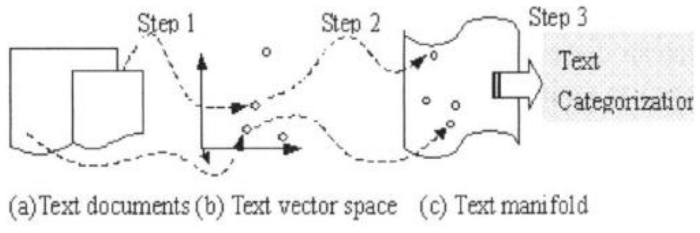


Figure 12. Framework of text categorization on text manifold [15]

Conclusions and future work

Typically, TF-IDF is used to discover semantic information from text, but the approach taken with this project is to apply to tokenize character values instead of words. Much of an HTML file may not have semantic meaning, such as an href reference link. By tokenizing each character, the algorithm can find useful information from elements which are not normally visible to the users of the websites. When starting this project, there was fear that there would not be any meaningful patters by vectoring characters instead of words. The results from the evaluation metrics used with this project suggests that this avenue is likely suitable to cybersecurity. While the evaluation results are fair and all to the left of the diagonal line with the ROC, expanding this project with proposed solution, such as using a text manifold (such as Isomap, Locally Linear Embedding, Modified Linear Embedding, Hessian Eigenmappening, etc.), ontology mapping, and parallel processing could likely render exceptional performance.

There are many lessons learned while working on this project. The one that I would say is most meaningful is that preprocessing can be very time consuming. This is useful to know when planning future projects. I have also learned many useful skills, such as web crawling and text mining. The initial goal was to analyze much more HTML tags and other elements, but I realized about midway through the project that it would not be possible, give the time constraint. After starting to collect information from websites, there were problems with memory storage and curse of dimensionality. Lastly, I have learned that finding active malicious websites and labeled data is very challenging and costly.

References

[1] D. Canali, M. Cova, G. Vigna, and C. Kruegel, “Prophiler: a fast filter for the large-scale detection of malicious web pages,” in *Proceedings of the 20th international conference on World wide web - WWW ’11*, Hyderabad, India, 2011, p. 197, doi: 10.1145/1963405.1963436.

[2] A. van Deursen, A. Mesbah, and A. Nederlof, “Crawl-based analysis of web applications: Prospects and challenges,” *Sci. Comput. Program.*, vol. 97, pp. 173–180, Jan. 2015, doi: 10.1016/j.scico.2014.09.005.

[3] C. Liu, Y. Sheng, Z. Wei, and Y.-Q. Yang, “Research of Text Classification Based on Improved TF-IDF Algorithm,” in *2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)*, Lanzhou, Aug. 2018, pp. 218–222, doi: 10.1109/IRCE.2018.8492945.

[4] R. Kosala and H. Blockeel, “Web mining research: a survey,” *ACM SIGKDD Explor. Newsl.*, vol. 2, no. 1, pp. 1–15, Jun. 2000, doi: 10.1145/360402.360406.

[5] Y. Zhao, Y. Qian, and C. Li, “Improved KNN text classification algorithm with MapReduce implementation,” in *2017 4th International Conference on Systems and Informatics (ICSAI)*, Hangzhou, Nov. 2017, pp. 1417–1422, doi: 10.1109/ICSAI.2017.8248509.

[6] S. Jiang, G. Pang, M. Wu, and L. Kuang, “An improved K-nearest-neighbor algorithm for text categorization,” *Expert Syst. Appl.*, vol. 39, no. 1, pp. 1503–1509, Jan. 2012, doi: 10.1016/j.eswa.2011.08.040.

[7] R. Cooley, B. Mobasher, and J. Srivastava, “Web mining: information and pattern discovery on the World Wide Web,” in *Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence*, Newport Beach, CA, USA, 1997, pp. 558–567, doi: 10.1109/TAI.1997.632303.

[8] M. Kumar, R. Bhatia, and D. Rattan, “A survey of Web crawlers for information retrieval,” *WIREs Data Min. Knowl. Discov.*, vol. 7, no. 6, Nov. 2017, doi: 10.1002/widm.1218.

[9] B. Trstenjak, S. Mikac, and D. Donko, “KNN with TF-IDF based Framework for Text Categorization,” *Procedia Eng.*, vol. 69, pp. 1356–1364, 2014, doi: 10.1016/j.proeng.2014.03.129.

[10] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Elsevier, 2012.

[11] V. Gupta and G. S. Lehal, “A Survey of Text Mining Techniques and Applications,” *J. Emerg. Technol. Web Intell.*, vol. 1, no. 1, pp. 60–76, Aug. 2009, doi: 10.4304/jetwi.1.1.60-76.

[12] Scrapy developers, “Scrapy Documentation,” Sep. 2020.

[13] J. N. Mandrekar, “Receiver Operating Characteristic Curve in Diagnostic Test Assessment,” *J. Thorac. Oncol.*, vol. 5, no. 9, pp. 1315–1316, Sep. 2010, doi: 10.1097/JTO.0b013e3181ec173d.

[14] A. Doan, J. Madhavan, P. Domingos, and A. Halevy, “Learning to Map between Ontologies on the Semantic Web,” *Proc. 11th Int. Conf. World Wide Web*, 2002.

[15] G. Wen, G. Chen, and L. Jiang, “Performing Text Categorization on Manifold,” in *2006 IEEE International Conference on Systems, Man and Cybernetics*, Taipei, Taiwan, Oct. 2006, pp. 3872–3877, doi: 10.1109/ICSMC.2006.384735.