RSGC
Royal St. George's College

The Young Researcher

2022 Volume 6 | Issue 1

Using Recurrent Neural Networks and Web Crawlers to Scrape Open Data from the Internet

Dhiraj Kanneganti

Recommended Citation
Kanneganti, D. (2022). Using recurrent neural networks and web crawlers to scrape open data from the Internet. *The Young Researcher*, 6(1), 60-71. http://www.theyoungresearcher.com/papers/kanneganti.pdf

ISSN: 2560-9815 (Print) 2560-9823 (Online) Journal homepage: http://www.theyoungresearcher.com

Recurrent Neural Networks and Web Crawlers

Dhiraj Kanneganti

Web crawling techniques in conjunction with Recurrent Neural Networks (RNNs) have been applied to several areas in the field of data mining on the Internet, but how they would best be applied to searching for open datasets has not yet been studied. Open data are data generated by the community and serve as effective alternatives to more centralized data collection, such as the US Census. Since the individuals and small organizations that collect open data often lack the infrastructure to make it widely available, open data portals serve as key access points to centralize open data. Unfortunately, due to lack of funding, open data portals struggle to efficiently scrape large amounts of open data from the internet. The purpose of this experiment is to bridge the gap between open data sources and open data portals by creating an algorithm that can quickly find open data on the internet. Through the use of an RNN and focused web crawler, a search algorithm was developed that could scrape 1000 web pages per minute and identify open datasets at an 85% accuracy, both metrics suggesting that the algorithm is a significant improvement over existing open data collection methods. For future research into this field of study, this work suggests that the application of automated open data collection and the implications of the proliferation of open data portals be studied.

Introduction

With the amount of information on the internet rapidly increasing over the past decade, the need for computer programs capable of filtering through online data, also known as web scraping algorithms, has significantly grown [1,2]. Web scraping algorithms have been implemented in order to handle large amounts of information in several data-dependent fields such as data storage, search engines, and medical databases [3]. In particular, one type of web scraping algorithm, known as the web crawler, has grown in popularity in recent years. The web crawler is a type of web scraping algorithm that functions by accessing web pages through hyperlinks inputted by the user to extract Hyper Text Markup Language (HTML) elements[1] from the web page. The web crawler will then access all the links on the web page and will keep repeating the cycle until the user terminates the program or the web crawler encounters a "dead end," which is a web page with no further links [2]. Web crawlers have experienced this surge in popularity over other types of web scraping algorithms due to their simplicity and ability to adapt to a wide variety of situations [3]. While there are dozens of web crawling techniques available, the most commonly used is the distributed web crawler, which is similar to that of the basic web crawler blueprint, except that the distributed web crawler is designed to extract large amounts of information from many different styles of web pages [4]. However, little research has been conducted on the other types of web crawling techniques, which has led to many questions

1. HTML elements refers to the semantic categories used in HTML code to group text together. Notable examples are "title," "header," etc.

about the effectiveness of other types of web crawling techniques that were not built to be a "one size fits all," like the distributed web crawler.

Much of the interest in web crawling has been fueled by potential applications in the data scraping market, which has led to a disproportionately heavy focus on distributed web crawling techniques over other forms of web crawlers [5]. However, there has been a recent spike in the usage of other web crawling techniques, specifically focused crawlers, which are web crawlers designed to search for and extract information from the internet [6]. Focused crawlers have a distinct advantage over distributed crawlers when it comes to finding specific information on the web; however, focused crawlers have taken the backseat to distributed crawlers since, unlike distributed crawlers, focused crawlers cannot sort through large amounts of information, which is the primary application of web crawling in the status quo [5,6]. Recent advances in data processing of focused crawlers have led to an increase in their viability in the data scraping market. Yet, the lack of research on focused crawlers has led to many struggles to find the most effective applications of this powerful technology. It is hoped that the results of this research will help pave the way for more insights into focused crawlers and their potential.

Literature Review

Thanks to the development of data processing technologies, focused web crawlers are well on their way to revolutionizing data retrieval on the open web. Focused web crawlers have been implemented in a variety of fields from hospital database administration to cybersecurity [7]. One prominent example of the use of focused crawlers was the creation of a web crawling bot by engineers at the Bharati Vidyapeeth College of Engineering capable of tracking price changes of a specific product across thousands of different e-commerce sites simultaneously [8]. The ability to extract and maintain access to that much information at once is practically unheard of in the current data market and is just one example of the many potential uses of focused web crawlers. And while e-commerce is a field which has seen a drastic rise in the use of focused crawlers, text-based web page filtration has seen an even more extreme rise. Text-based filtering is when focused crawlers are trained to look for certain features in the text of certain HTML elements of web pages [10]. Focused crawlers play a key role in this process. In a text-based filtering project conducted by Gunawan et al., researchers of computer science at the University of Sumatra, found that "focused web crawler[s] … were critical to improve performance of … the automatic harvesting of articles from online publications" [10]. Not only are focused crawlers key to text-based classification in the status quo, but Dong et al., researchers of web design at Curtin University of Technology, also predict that "the plethora of data on the internet will force … increased usage of focused web crawlers in the future" [11].

Text-based filtering does not solely comprise the focused crawler; it also requires the use of a Recurrent Neural Network (RNN), which is a type of machine learning algorithm2 that creates a network of information that classifies input data by inputting it into layers of nodes, depicted as vertical columns of circles in Figure 1, which are connected by continuous series of loops, depicted as the lines in between each layer in Figure 1, known as neural networks [10, 11]. RNNs can be used in conjunction with focused crawlers in text-based filtration due to their ability to classify text through a process known as tokenization. RNNs use tokenization to convert textual data, which would consist of the HTML elements extracted from web pages using the focused crawler, into numerical data referred to as "tokens" [12]. These tokens are inputted into the RNN and are classified into categories based on underlying characteristics identified by the neural network. Due to the high classification accuracy of tokenization, RNNs have become the industry standard of machine learning classifiers to be used in conjunction with focused classification in text-based classification. Zulqarnain et al, researchers of artificial intelligence at the University of Tun Hussein, conducted an experiment on the text classification accu-

2. Machine learning is a subset of computer programs that can find trends in data that are difficult to find [13]

racy of several machine learning models and found that "RNNs … have been achieving outstanding results in various areas of … text classification" [13].

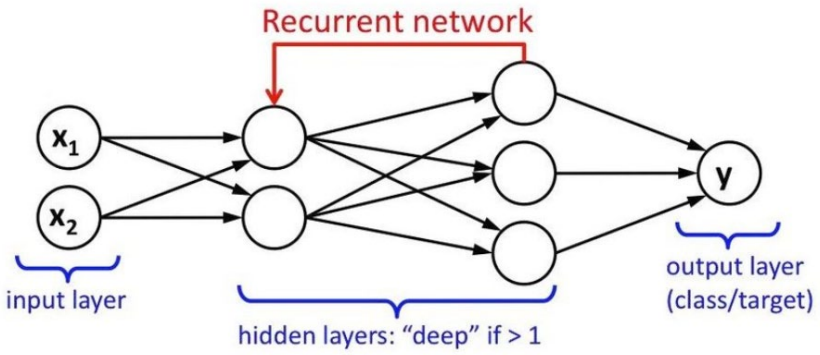

Figure 1 | Neural Network Neural networks are a series of layers that have checkpoints called "nodes." The data will activate a node if it meets certain criteria set by the neural network, which can then classify data based on the path of activated nodes. In this illustration, the circles represent nodes and the vertical columns of nodes represent the layers [14].

Many studies have combined the usage of RNNs and focused crawlers in order to make text-based web page classification systems. Buber and Diri, researchers of computer science at Yildiz Technical University, created "an information retrieval application that… makes use of an RNN … based architecture in conjunction with… a focused web crawler… which achieved a web page classification accuracy of 85%" [15]. The synergy between RNNs and focused crawlers has led to successful applications created in both the scholarly and corporate fields.

The research conducted in this project is valuable for a multitude of reasons. Not only will the results of this experiment give important insight into the potential uses of focused crawlers, but they will also generate a tool that could be used to extract open data. The goal of this experiment is to create a program that uses Recurrent Neural Networks in conjunction with Web Crawling techniques to crawl through the internet and identify open data to be scraped.

Open data, also referred to as community-generated datasets, are collected from the internet independently from large institutions[16]. Currently, there is a strong need in the status quo for community-generated datasets. The current source of data for many businesses and individuals is large government datasets like the US Census. However, the Census is oftentimes unreliable due to data collection occurring once every ten years [17]. Furthermore, a lack of government funding in less populated areas means that the Census often produces inaccurate data about rural communities [18]. However, there is a solution to the lack of reliable data in the form of open data portals. Open data portals are websites that keep data that is generated by the community. Open data gives rural populations an opportunity to collect their own data, which, in theory, would be more accurate than that collected by an infrequent, centralized survey. Yet, there is often a lack of communication between open data portals and those who make community-generated datasets. The current expansion process used by open data portals is to manually scour the internet for these community-generated datasets [19]. This is because community-generated datasets are created by a wide variety of individuals, so it would be incredibly difficult to contact each individual and ask for their data. However, the process of combing through the internet, while more effective than directly asking the creator(s) of the dataset, is also very time consuming and has stunted the growth of open data portals around the world.

Even though text-based classifiers created to find open data on the internet would be ideal for alleviating this issue, the lack of profitability of open data portals has led to a gap in the research in this field due to a lack of financial incentive. As Zuiderwijk & Janssen, researchers at the Delft University point out, there is "little research on current methods open data portals could use to expand their databases" [16]. This leads to the research question: How can Recurrent Neural Networks be used in conjunction with Web Crawling techniques to identify and scrape open datasets? Thus, by creating a program which can automatically search the internet for open data, this gap can be alleviated, and it could spark the proliferation of open data portals around the world [16].

Methodology

The purpose of this experiment is to create an application capable of making community-generated datasets more accessible to open data portals. When building this program, there was an emphasis placed on efficiency and usability. In order to prioritize these two aspects, an Engineering Design methodology was developed to create and test an application capable of crawling through the internet in order to extract community-generated data.

A. Development Environment & Software Packages

To first go about constructing such an application, one must consider the software that will be used to construct the program. Although there are many possible coding languages that could be utilized in this experiment, Python version 3.6 was selected due to its flexibility and the availability of plenty of up-to-date machine learning libraries [20]. Python 3.6 is one of the latest versions of the language and is compatible with the web scraping software used in this experiment. Older versions of Python often have severe issues when running web scraping software, and were avoided in this project to prevent unnecessary bugs and/or glitches. PyCharm was selected as the primary development environment for this project as PyCharm is one of the premier integrated development environments[3] (IDEs) for Python. And while there are several other IDEs that were considered, such as Google CoLab and Sublime Text Editor, Pycharm was chosen due to its compatibility with the wide variety of software packages[4] needed in this experiment. Shetty et al, esteemed researchers of machine learning at the Anjuman Institute of Technology who have had significant experience with several Python IDEs, explain:

"Pycharm allows for complex… manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with [other] programs… making it a useful tool in … software experimentation" [21].

PyCharm enables the researcher to have flexibility when creating the application and the ability to approach this issue from multiple angles, both of which are valuable tools in this experiment.

Two main software packages were installed in order to create the application, ScraPy and TensorFlow. ScraPy is a web scraping framework that comes with pre-built web crawlers called "spiders" that can be modified to fit the user's needs. ScraPy is not the only web scraping package available in Python 3.6, but it was used in this project due to its "powerful engine capable of controlling large amounts of data… and its ability to create stable focused crawlers" [22]. Other web crawling packages such as BeautifulSoup and Selenium often struggle when scraping from a variety of different web pages, but ScraPy's data scraping engine is perfectly suited for doing such. TensorFlow was also used because it is the only software package available that has all of the tools required by the researcher to construct the RNN [22]. Other minor packages that were installed for machine learning include Keras, Pandas, NumPy, and Sklearn.

B. Web Crawler Construction

The web crawler was constructed in PyCharm using the ScraPy framework. The "broad crawl" spider was selected and installed from the ScraPy library as it is the most suitable to create the desired focused crawler. The web crawler was then adapted to scrape HTML elements of the previously selected web pages such as the title, paragraphs, and headers. These three HTML tags were chosen in particular since they are the most indicative of a webpage's genre and purpose [9]. For the RNN to determine whether or not a web page is a community-generated dataset, it is important that it has access to the most important information on the web page, which most often lies within these three categories.

3. An IDE is a software that can be used as an environment to code computer programs.

4. A software package is a pre-made assortment of code that can be used to make new coding projects. Due to Python's intertwined development with machine learning, many of Python packages relate to AI.

C. RNN Construction

The first step in the construction of the recurrent neural network is to gather web pages that can be used to test the web crawler and train the RNN text classifier. In this experiment, 300 web pages were collected after evaluation of the amount of text, size, and genre of the web page, a similar selection framework to that used for Zulqarnain et al.'s experiment. One hundred of the web pages can be considered community-generated datasets. This 33% split between non datasets and datasets was chosen because it is considered the industry standard in most RNN test classification projects [15]. It has been found that the most optimal way to train an RNN text classifier is to have one third of the training data contain the specific attribute that is being classified [9,10,12]. Training data is used to construct the machine learning model, while testing data can be used to check the accuracy and refine the trained model. There was also an effort to select web pages from a variety of different genres: business, entertainment, environment, demographics, and culture. This would ensure that the RNN would have a wide variety of text to train on, which would boost its accuracy and robustness [15]. Each web page's genre would not be included in the data analyzed by the RNN; however, the genre was used for analysis of the RNN's accuracy, which will be later discussed.

After the web crawler is able to scrape the desired text from the web pages, the RNN text classifier must be built. This was done through the use of TensorFlow and other previously mentioned machine learning packages. The tokenization process was performed through the Natural Language ToolKit (NLTK). The NLTK is one of the most prominent software libraries used in tokenization for RNNs because it can be applied to a wide variety of RNNs and the implementation process of the NLTK is very simple and quick. After the tokenization and RNN have been constructed, the text classifier must be trained on the datasets that were selected. Since the RNN's accuracy has to be tested later on using new web pages it has not seen before, the web pages must be split into training and testing. The training group will contain about 66% of the web pages and the testing group will consist of the remaining 33%. The groups were randomly chosen and the training dataset, which consists of text from the titles, paragraphs, and headers of the web pages in the training group, was automatically inputted into the tokenization process and RNN. The final step in the construction of the RNN is the creation of the output layer. This layer will have two nodes: open dataset and not an open dataset. The text classifier's analysis of the scraped text will determine which of these two nodes will be activated, outputting the classification of the web page.

After training the RNN and constructing the web crawler, the final step before evaluation is to connect the two components. This can be done by having the web crawler automatically dump scraped text into a vector array5 which the RNN will access and classify, both of which will occur in real time.

D. CSV File Intermediary

Originally, the scraped text was immediately entered into the RNN, however, this led to several issues with accuracy since much of the unstored text would be lost. This lack of text drastically reduced the RNN accuracy, as it did not have sufficient information to construct the classification model. To better store information, the scraped text would be sent to a Comma Separated Values (CSV) file, where it could be stored without data loss and easily accessed by the RNN. Storing data in a CSV file meant that a NumPy function had to be used to process the CSV data to turn it into tokens. The use of this converter led to several latency issues and required several safeguards to be coded in to ensure that glitches with the CSV would not break the program.

5. A vector array is a two-dimensional storage unit that can be used to hold data from multiple variables. In the context of this experiment, the vector array will hold four columns of data: header, paragraphs, titles, and 1 or 0 to indicate if the web page is an open dataset.

E. Evaluation

In order to evaluate the effectiveness of the program, criteria that measure speed and accuracy such as web pages scraped per minute (WSPM) and RNN text classification accuracy of web pages were considered.

WSPM was used to measure the usability of the web crawling algorithm. If the WSPM is considered too slow or subpar, no open data portal will be incentivized to use this program since it is not an improvement over manual collection of community-generated datasets [16] . It is crucial that the speed of this program is a considerable improvement over current collection methods, otherwise the long-term project goal of increasing the proliferation of open data portals will not be met.

Classification accuracy is also held in high importance because speed of text extraction is irrelevant if the program cannot consistently identify community-generated datasets. The goal of this project is to have an RNN accuracy of at least 80%. Buber and Diri claim that RNN text classifiers that have at least an 80% accuracy can be deemed "viable tools in most industries" [15]. RNN Accuracy will be measured by comparing the RNN's classification of a web page to the web page's "ground truth." Each web page used in this experiment was manually evaluated by the researcher and determined to be an open dataset or non-open dataset. This manual classification is a web page's "ground truth." The RNN will attempt to determine which of these two categories a web page falls into after analyzing the scraped text from that web page. The RNN's classification for each web page will then be compared to the "ground truth." If the RNN's classification is the same as the "ground truth," the classification is considered to be successful; if it differs, then the classification will be unsuccessful. The number of successes will be divided by the total number of attempts to determine the classification accuracy.

In addition to accuracy, loss will also be used to measure the effectiveness of the RNN. Loss is a self-inflicted penalty used by machine learning models to determine their efficacy [15]. A higher loss indicates a weaker performance. Loss is a reliable indicator of the model's performance and will be used to evaluate the RNN in conjunction with accuracy.

F. Optimization

After the accuracy and loss are obtained, the RNN will then be optimized using epochs. An epoch is an optimization the RNN makes as it re-analyzes data to create an improved model. Epochs allow the algorithm to make corrections to faulty predictions and find new trends in the data. The RNN was programmed to re-adjust its internal parameters after every epoch in order to decrease loss as much as possible. In this experiment, 10 epochs were performed to optimize the RNN. This specific number of epochs was selected as too many epochs would result in the RNN "over-learning" which the process of the algorithm becoming too accustomed to the training dataset and unable to classify any new data, and too few epochs would not give the algorithm enough opportunities to adjust itself.

Results

After the algorithm was executed in a PyCharm environment, the loss and accuracy were automatically measured using internal functions pre-built into the Keras software package. Figure 2 and Figure 3 indicate the accuracy and loss of the RNN when classifying web pages in Training and Testing Datasets, respectively. The accuracy and loss are graphed over the number of epochs. As such, in both datasets, the classification accuracy of the RNN increases as epochs increase, while the loss decreases. These values are also available in the Appendix.

In order to more closely examine the performance of the model and put the RNN's accuracy and loss into context, confusion matrices were generated for both the Training and Testing datasets. Figures 4 and 5 are confusion matrices that visually display Predicted Positives, Predicted Negatives, True Positives, and True Negatives. In this study, a Predicted Positive is a web page that the RNN classifies as an open dataset. A Predicted Negative is a web page that the RNN classifies as not an open dataset. A True Positive is a web page that is an open dataset, and a True Negative is a web page that is not an open dataset. Of the 300 webpages used to construct and train the RNN, 262 were classified correctly, bringing the classification accuracy to 87.33%. However, when just analyzing the results of the testing dataset, the model had a classification accuracy of 85%. This is the metric that will be used for evaluation, since the training dataset predictions may be slightly biased since they were used to construct and test the uncompleted versions of the model, while the testing dataset was only used to test the final version. Therefore, when analyzing the final version of the RNN classifier, the testing dataset accuracy is more representative.

Furthermore, the classification accuracy was also further analyzed using the genres of the web pages. The percentage of correctly classified web pages in each genre was plotted in Figure 6.

Metrics to evaluate the performance of the web crawler were also collected, mainly the WSPM. The WSPM was automatically measured using functions from the ScraPy framework and came out to be 1000 web pages per minute.

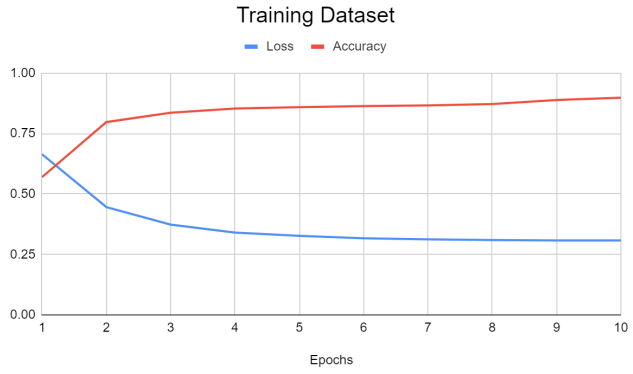

Figure 2 | Accuracy and Loss of RNN classifying Training Dataset over 10 Epochs

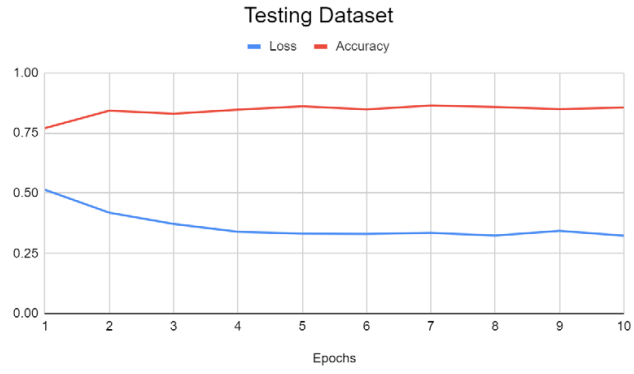

Figure 3 | Accuracy and Loss of RNN classifying Testing Dataset over 10 Epochs

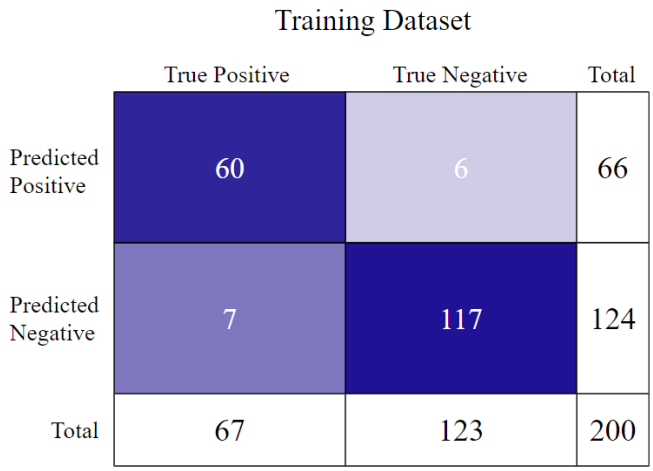

Figure 4 | Confusion Matrix of Training Dataset Classification

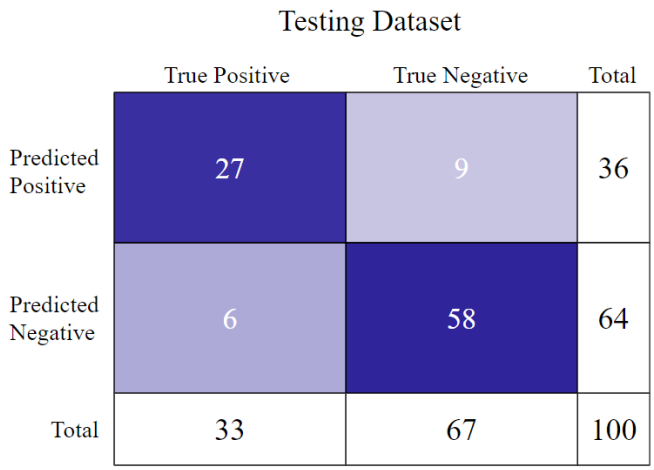

Figure 5 | Confusion Matrix of Testing Dataset Classification

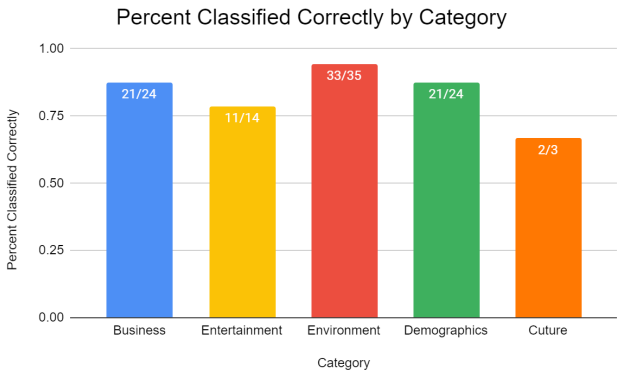

Figure 6 | Classification Accuracy of Different Open Dataset Genres At the top of each bar, the number of correctly classified datasets is divided by the number of datasets within the genre.

Discussion

The results of this experiment suggest there is important discussion to be had about the use of RNNs to identify and extract open datasets from the internet. The discussion has been split into the following sections: Web Crawling, RNN Performance, Classification, and Web Page Genres.

A. Web Crawling

One of the primary components of this experiment was to evaluate the ability of a focused crawler to scrape text from open datasets. As previously mentioned, the WSPM of the web crawler was measured to be 1000, which is significantly higher than existing manual open dataset collection methods, meeting one of the research objectives. However, when compared to other crawlers, the classification algorithm is somewhat slow. Botify, one of the leading companies in the web crawling space has produced crawlers that can scrape text at 12000 WSPM [23]. This drastic difference in speeds is most likely because the crawler in this experiment must wait for the RNN to classify the web page before moving to the next page, while the Botify crawler is not attached to any external classification algorithm.

While the crawler was efficient at scraping text from most web pages, it did run into several issues with more complicated web pages. The crawler functions by looking for HTML elements within the web page; if they are present then the crawler will extract the text and send it to the RNN, if not, the crawler will move onto the next page. Embedded HTML elements are elements inside of another element. The crawler aimed to scrape the paragraphs, titles, and headers of

each web page, but it was observed after preliminary testing that these target HTML elements that were embedded inside non-target elements would be completely skipped. For example, the text in a header embedded in a footer would not be extracted. This was not a severe issue as the amount of text left unextracted had a minimal impact on the RNN classification accuracy. The real issue was with non-target elements embedded inside target elements. In these cases, the crawler would instantly shutdown and lead to several code-breaking errors. While this occurred merely four times during the hundreds of tests, it is indicative of the primary weakness of the focused crawler: inflexibility.

The distinction between distributed and focused crawlers is that distributed crawlers are much more open-ended and adaptable than their counterparts [4, 21]. Adaptability comes at a cost, as distributed crawlers have a significantly longer run time than focused crawlers [4]. This trade-off between time consumption and flexibility was the primary reason for the use of focused crawlers in this experiment, as it was hypothesized that the rigidity of focused crawlers would not affect their ability to find open datasets. There is no need for adaptability when the crawler is looking for only one type of web page. However, it was apparent when the focused crawler was struggling to scrape text from complex pages with embedded elements that there were drawbacks to using the focused crawler. The straightforwardness of the focused crawler, while increasing time efficiency and helping meet one of the aims of this research, led to several errors when the crawler could not immediately find what it was looking for. This presents a unique scenario, as the crawler could be modified to scrape all the text from the web page, but that would take significantly more time, mostly removing the advantage of using focused crawlers. Through the implementation of focused crawlers in complex scraping tasks, this experiment has brought new insights into the tradeoff between time consumption and web scraping efficiency of focused crawlers.

B. RNN Performance

After epoch optimization, the RNN had a web page classification accuracy of 85% on the testing dataset, meeting the research goal of above 80% accuracy. This means that the RNN can be considered a viable tool that can be used within professional environments, including open data portals [15]. The loss of the RNN was also very low, at only .32 for the testing dataset, further indicating that the RNN had high efficacy. Even though the accuracy of the RNN was satisfactory, there were some issues with its performance. RNNs usually classify text with no time constraints; however, in this experiment, the RNN had to classify a web page before the crawler could move to the next, meaning that the latency issues caused by the CSV converter were a significant issue. The implementation of a CSV file to act as a real time medium gave useful information to the potential data loss and lag with real-time text classification.

C. Classification

Interesting trends emerged when analyzing the classification patterns of the RNN. In both the Training and Testing datasets, the accuracy of the RNN when classifying non-open datasets was higher than when classifying open datasets. The classification accuracy for True Negatives in the Training dataset was 95%, while it was 89% for True Positives. In the Testing dataset, the classification accuracy was 86%, but only 81% for True Positives. This difference between True Positives and True Negatives may have been caused by non-open data web pages that contained datasets. When composing the 300 web pages for this experiment, several online datasets that are not open datasets, such as census data, were selected. This was done to ensure the RNN would not create a model that determined if a web page was an open dataset solely based on the presence of numbers or tables.

One recommendation the researcher makes for future work is that an RNN trained to classify open datasets is compared with a standard algorithm that determines whether or not a web page is a dataset by looking for the presence of numerical data, and the absence of keywords such as “government,” “census,” etc. The lack of an RNN would drastically decrease the time needed to execute the program, and if the classification accuracy of the standard algorithm is similar, then it could provide new insights into the application of RNNs.

D. Web Page Genres

The classification accuracy of the RNN was also measured across the different genres of web pages in the Training dataset. Across the five genres selected, Environment web pages were the most likely to be correctly classified at 88%. This was most likely because environmental open datasets are easily distinguishable due to their frequent use of data tables. Similar arguments can be made about Business and Demographics web pages which both had the second highest accuracy at 87.5%.

E. Limitations

While the results of this experiment are promising for the proliferation of open data portals, the results come with a caveat. There were two main limitations to this experiment: the RNN could only classify datasets in English and the composition of the web pages chosen may have influenced the classification accuracy. Even though the NLTK software package can be easily modified to be compatible with dozens of languages, this RNN could only classify English web pages [13]. This was due to the fact that the English-speaking researcher had to verify the web pages the RNN trained on, meaning that only English web pages were selected for this experiment. Further research should be conducted to examine if the classification accuracy of the RNN changes drastically as the language of the web pages changes. Secondly, the uneven distribution of genres may have led to alterations in the RNN classification model. Since the RNN was largely successful across all types of genres and this distribution is similar to what the RNN would experience in a non-controlled environment, this should not have too large of an impact on the results [15]. The researcher suggests that experimentation should be conducted, similar to that of Biber and Diri, that analyzes the effect of different genres in the Training set on classification accuracy.

Conclusion

The purpose of this experiment was to create a real time text classification system using an RNN and web crawler to identify open data sets on the internet. The algorithm's drastic improvements in speed and accuracy over existing methods to find open data suggest that this purpose was filled. Buber and Diri's framework of evaluating RNNs argues that an 80% classification accuracy is critical to be a useful tool, a metric that the RNN of this experiment comfortably meets [15]. These overall positive results validate several previous experiments conducted in the fields of web crawlers and RNN text classification.

The speed and general efficiency of the focused crawler provide further evidence to substantiate Dong et al.'s argument that focused crawlers will see a drastic increase in usage in the future [10]. Currently, outside of the focused crawler, there are no other tools in the market that are capable of filtering through that amount of text so quickly. The ability of the focused crawler to swiftly parse through HTML elements is unmatched and was key to meeting the experimental goal of improving upon existing open data collection techniques.

As previously mentioned, the primary shortcoming of the algorithm was the focused web crawler's inability to effectively scrape data from more complicated web pages, which reaffirms the results of Shkapenyuk & Suel who find that distributed crawlers are more suited to scraping data from more complicated web pages [4]. Yet, it contradicts the work of Khan & Sharma who claim that focused crawlers can be applied to a wide variety of situations [3]. The disagreement between studies and the results of this experiment defines a niche for focused web crawlers that entails a distinct usage pattern, only retrieving data from simplistic web pages, but a usage pattern that has usefulness across a wide range of industries. After analyzing the effectiveness of the algorithm, this experiment creates a new understanding of the strengths of focused web crawling algorithms with their high scraping speed, as well as the potential issues focused crawlers may have with more complicated web pages. The results of the RNN also paint a new understanding of their high accuracy with text classification of open datasets, a novel application of such text classifiers, but also the difficult implemen-

tation and high maintenance required to ensure the RNNs work properly.

While the experimental goals of this research have been met, further study needs to be done on the actual implementation of this algorithm into open data portals. Studies such as Pisani et al. & Zuiderwijk & Janssen give valuable data about the impacts of the proliferation of open data portals, but existing methods to collect open data have prevented this proliferation from occurring in the first place. In order to better evaluate the new understanding generated from this experiment, as well as garner more insight into the nature of open data and the effects of open data portals on society at large, the implementation of this algorithm in the future is crucial.

References

[1] A. Holst, "Total Data Volume Worldwide 2010-2025," Statista, 07-Jun-2021. [Online]. Available: https://www.statista.com/statistics/871513/worldwide-data-created/.

[2] T. Karthikeyan, K. Sekaran, D. Ranjith, V. Vinoth Kumar, and J. M. Balajee, "Personalized content extraction and text classification using effective web scraping techniques," International Journal of Web Portals, vol. 11, no. 2, pp. 41–52, Jul. 2019. Available: https://www.researchgate.net/publication/336733112_Personalized_Content_Extraction_and_Text_Classification_Using_Effective_Web_Scraping_Techniques

[3] S. Sirisuriya, "A Comparative Study on Web Scraping," International Research Conference KDU, vol. 8, no. 1, pp. 135–140, Nov. 2015. Available: http://ir.kdu.ac.lk/bitstream/handle/345/1051/com-059.pdf?sequence=1&isAllowed=y

[4] V. Shkapenyuk and T. Suel, "Design and implementation of a high-performance distributed web Crawler," International Conference on Data Engineering, vol. 18, no. 1, pp. 1–12, 2012. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.13.4762&rep=rep1&type=pdf

[5] L. Yu, Y. Li, Q. Zeng, Y. Sun, Y. Bian, and W. He, "Summary of web crawler technology research," Journal of Physics: Conference Series, vol. 1449, no. 1, pp. 1–7, Jan. 2020. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1449/1/012036/pdf

[6] T. V. Udapure, R. D. Kale, and R. C. Dharmik, "Study of web crawler and its different types," IOSR Journal of Computer Engineering, vol. 16, no. 1, pp. 01–05, Feb. 2014.

[7] M. A. Khan and D. K. Sharma, "Self-adaptive ontology-based focused crawling: A literature survey," 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), vol. 5, no. 1, pp. 595–601, Sep. 2016. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7785024

[8] A. Ambre, P. Gaikwad, K. Pawar, and V. Patil, "Web and Android application for comparison of e-commerce products," International Journal of Advanced Engineering, Management and Science, vol. 5, no. 4, pp. 266–268, Apr. 2019.

[9] F. Brenner, F. Platzer, and M. Steinebach, "Discovery of single-vendor marketplace operators in the Tor-Network," *The 16th International Conference on Availability, Reliability and Security*, 2021.

[10] D. Gunawan, A. Amalia, and A. Najwan, "Improving data collection on article clustering by using distributed focused crawler," Data Science: Journal of Computing and Applied Informatics, vol. 1, no. 1, pp. 1–12, Jan. 2017. Available: https://www.researchgate.net/publication/330187644_Improving_Data_Collection_on_Article_Clustering_by_Using_Distributed_Focused_Crawler

[11] H. Dong, F. K. Hussain, and E. Chang, "State of the art in semantic focused crawlers," Computational Science and Its Applications – ICCSA 2009, pp. 910–924, Dec. 2009. Available: https://www.researchgate.net/publication/44241179_State_of_the_Art_in_Semantic_Focused_Crawlers

[12] L. Yu, Y. Li, Q. Zeng, Y. Sun, Y. Bian, and W. He, "Summary of web crawler technology research," Journal of Physics: Conference Series, vol. 1449, no. 1, pp. 1–7, Jun. 2020.

[13] M. Zulqarnain, R. Ghazali, Y. M. Hassim, and M. Rehan, "A comparative review on Deep Learning Models for text classification," Indonesian Journal of Electrical Engineering and Computer Science, vol. 19, no. 1, pp. 325–335, Jul. 2020. Available: https://www.researchgate.net/publication/340601688_A_comparative_review_on_deep_learning_models_for_text_classification

[14] V. Mishra, M. Agarwal, and N. Puri, "Comprehensive and comparative analysis of neural network," *International Journal of Computer Applications*, vol. 2, no. 8, pp. 126–137, 2018.

[15] E. Buber and B. Diri, "Web page classification using RNN," Procedia Computer Science, vol. 154, no. 1, pp. 62–72, Jan. 2019. Available: https://www.researchgate.net/publication/334474452_Web_Page_Classification_Using_RNN

[16] A. Zuiderwijk and M. Janssen, "The negative effects of open government data - investigating the Dark Side of Open Data," Proceedings of the 15th Annual International Conference on Digital Government Research - dg.o '14, vol. 3, no. 2, pp. 147–152, 2014.

[17] S. Noble, "Does census data still have value almost ten years later?," OCSI, 17-Dec-2020. [Online]. Available: https://ocsi.uk/2020/11/23/does-census-data-still-have-value-almost-ten-years-later/.

[18] N. A. Wardrop, W. C. Jochem, T. J. Bird, H. R. Chamberlain, D. Clarke, D. Kerr, L. Bengtsson, S. Juran, V. Seaman, and A. J. Tatem, "Spatially disaggregated population estimates in the absence of national population and Housing Census Data," Proceedings of the National Academy of Sciences, vol. 115, no. 14, pp. 3529–3537, Feb. 2018. Available: https://www.pnas.org/content/115/14/3529

[19] J. Umbrich, S. Neumaier, and A. Polleres, "Quality Assessment and evolution of Open Data Portals," 2015 3rd International Conference on Future Internet of Things and Cloud, vol. 7, no. 3, pp. 1–8, Feb. 2015. Available: https://aic.ai.wu.ac.at/~polleres/publications/umbr-etal-2015OBD.pdf

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, and B. Thirion, "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research , vol. 12, pp. 2825–2830, 2011. Available: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?ref=https://githubhelp.com

[21] A. R. Shetty, F. B. Ahmed, and V. M. Naik, "CKD Prediction Using Data Mining Technique As SVM And KNN With Pycharm," International Research Journal of Engineering and Technology, vol. 6, no. 5, pp. 4399–4405, May 2019.

[22] Y. Fan, "Design and implementation of distributed crawler system based on Scrapy," IOP Conference Series: Earth and Environmental Science, vol. 108, p. 042086, 2018. Available: https://iopscience.iop.org/article/10.1088/1755-1315/108/4/042086/pdf

[23] A. Bouard, "Crawl speed: How many pages/second? 7 points to take into account," *Botify*, 11-May-2020. [Online]. Available: https://www.botify.com/blog/crawler-impact-performance. [Accessed: 23-Jun-2022].

Appendix

Appendix A | Loss and Accuracy of Training Dataset

Training Dataset		
Epoch	Loss	Accuracy
1	0.6647	0.5696
2	0.4459	0.7974
3	0.3735	0.8358
4	0.3404	0.8536
5	0.3268	0.8588
6	0.3167	0.8637
7	0.3126	0.8661
8	0.3094	0.8719
9	0.3081	0.8887
10	0.3079	0.8984

Appendix B | Loss and Accuracy of Testing Dataset

Testing Dataset		
Epoch	Loss	Accuracy
1	0.5139	0.7703
2	0.4192	0.8425
3	0.3722	0.8297
4	0.3396	0.8469
5	0.332	0.8604
6	0.331	0.8479
7	0.3355	0.8641
8	0.3239	0.8578
9	0.3438	0.849
10	0.3233	0.8557