EC504 Spring 2022 Project Report

Rapid Searching for Multiple Patterns in Large Texts Using Aho-Corasick Algorithm

Submission Date : 4 May 2022

Team Information

Jaeyoon Chung
SCC Username: jcdino
BUID: U40934352

Piyusha Dongre
SCC Username: pddongre
BUID: U21189658

Arnaud Harmange
SCC Username: arnaudh
BUID: U13657431

Farbin Fayza
SCC Username: ffayza
BUID: U13317302

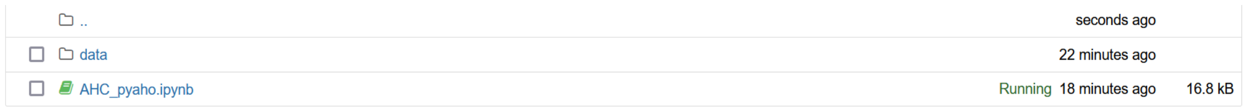
Chelsea Lau
SCC Username: lauche
BUID: U51375810

Abstract

The objective of this project is to implement and expand on the capabilities of the Aho Corasick algorithm that searches large bodies of text for patterns through the construction and traversal of tries. Extra capabilities include the option to enable or disable case sensitivity and wildcard search options. In this case, a "wildcard search" entails being able to search for patterns with an asterisk (*) that can be substituted for any character. The successful execution of these features as well as of the basic implementation has been validated through testing, the results of which will be analyzed in the report to come. Text from an article on rainbows as well as from "Harry Potter and the Sorcerer's Stone" were used as input files. The project was implemented using Python and Jupyter Notebook.

Instructions

This codebase is extremely simple to use. The code is entirely within a Jupyter notebook, which makes setting up and running the code straightforward. Once all of the files are downloaded from the project GitHub or SCC directory, navigate to the directory with all the files on the local machine through a terminal window. The only files needed are "AHC_pyaho.ipynb" and the two input files, "HP_1.txt" and "rainbow.txt" inside of the "data" folder (shown in Figure 1).



	Name	Status	Last Modified	File size
	..		seconds ago	
☐	data		22 minutes ago	
☐	AHC_pyaho.ipynb	Running	18 minutes ago	16.8 kB

Figure 1

Then run the "Jupyter Notebook" command in the terminal window and the project will be opened in a browser window. If the user does not have a Jupyter notebook installed, this can easily be done using the "pip install notebook" command. In the browser window, click on the jupyter folder, and then click on the AHC_pyaho.ipynb file to open the python notebook. Once the notebook opens in a new tab, the only steps required to run the code is to click the "Run" button at the top of the page (shown in Figure 2)

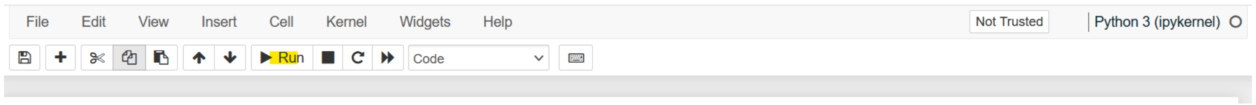
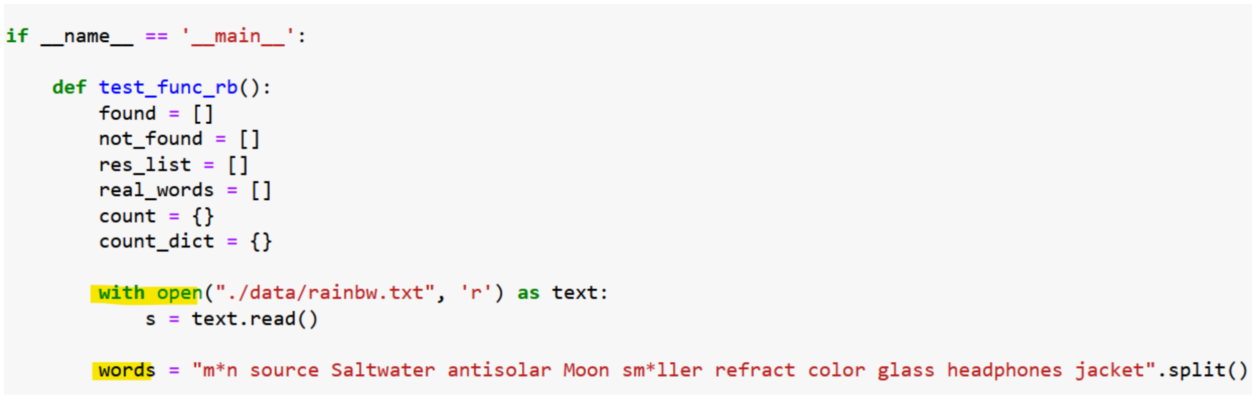


Figure 2

This will run the first test function defined at the bottom of the script. Clicking the "Run" button a second time will run the second test function. The first test function utilizes a smaller text about rainbows (rainbw.txt) while the second test function utilizes a much larger body of text, in this case the entire first Harry Potter novel (HP_1.txt).

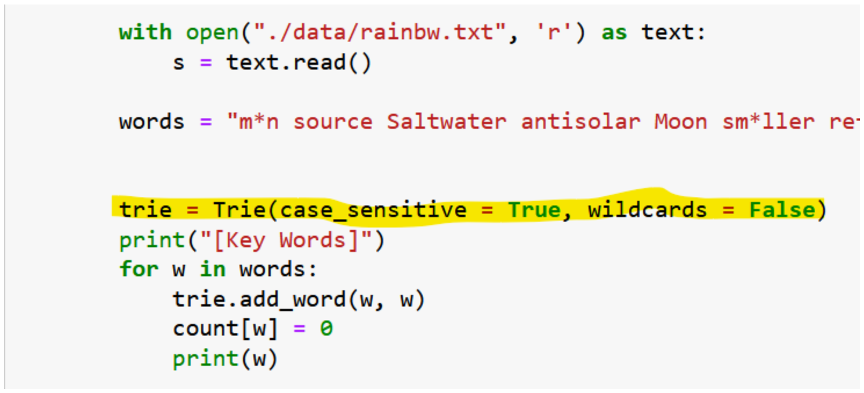
Both of these functions can easily be modified in order to run on other bodies of texts, as long as they are in txt format. All that must be done to change the text being used is changing the file path in the “With open” line in the code. The words to be searched for can also be modified by changing the values in the ‘words’ string in the code (shown in Figure 3). Words should be separated by a single space in order to ensure proper search functionality.



```python
if __name__ == '__main__':

    def test_func_rb():
        found = []
        not_found = []
        res_list = []
        real_words = []
        count = {}
        count_dict = {}

        with open("./data/rainbw.txt", 'r') as text:
            s = text.read()

        words = "m*n source Saltwater antisolar Moon sm*ller refract color glass headphones jacket".split()
```

Figure 3

The line of code that follows the declaration of search words is the initialization of the trie, which is where flags can be set for case sensitivity and wildcards. The case_sensitive flag allows for case sensitive search on the text. When set to True, the search is case sensitive, and when set to false, the algorithm will find all words that match the specified characters, regardless of case. This flag has a default value of False. The wildcards flag determines whether or not the algorithm will search for words with wildcards. When set to False, the algorithm will not use wildcard search, but when set to True the algorithm will have wildcard pattern matching enabled (shown in Figure 4).



```python
with open("./data/rainbw.txt", 'r') as text:
    s = text.read()

words = "m*n source Saltwater antisolar Moon sm*ller re


trie = Trie(case_sensitive = True, wildcards = False)
print("[Key Words]")
for w in words:
    trie.add_word(w, w)
    count[w] = 0
    print(w)
```

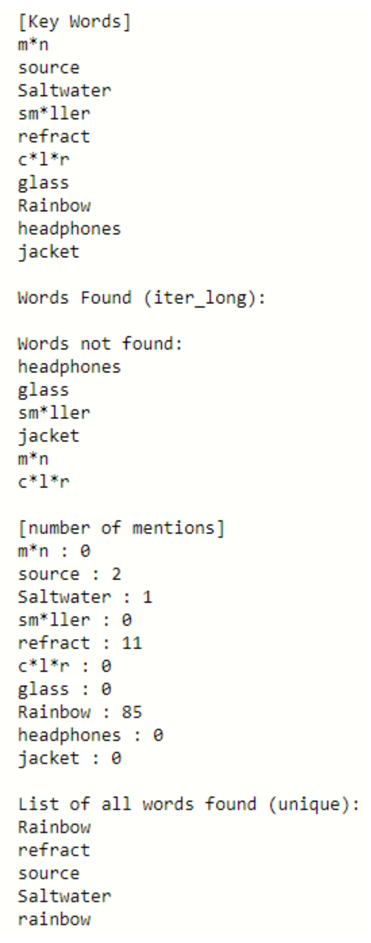
Figure 4

In order to specify a wildcard, in the word declaration, use an asterisk (*) to indicate a wildcard character. For example, in order to search for all three letter words starting with the letter “m” and ending with the letter “n”, the user should input “m*n” as a word in the ‘words’ string. It should be noted that it is not possible to search for words with a wildcard as the first character, since it is not able to correctly start traversing the Trie. All other positions for the wildcard character are valid.

The rest of the code that in the testing function first adds all of the words to be searched for to the trie, then initializes the Aho-Corasick automaton, and then removes any special characters from the text being searched. The remaining code simply takes the results from the search and formats it in such a way that it is easily readable by the user. The output specifies the key words being searched for, the words found in the text, number of occurrences of each word found in the text, and a list of all the unique words found in the text.

Method of Output Validation

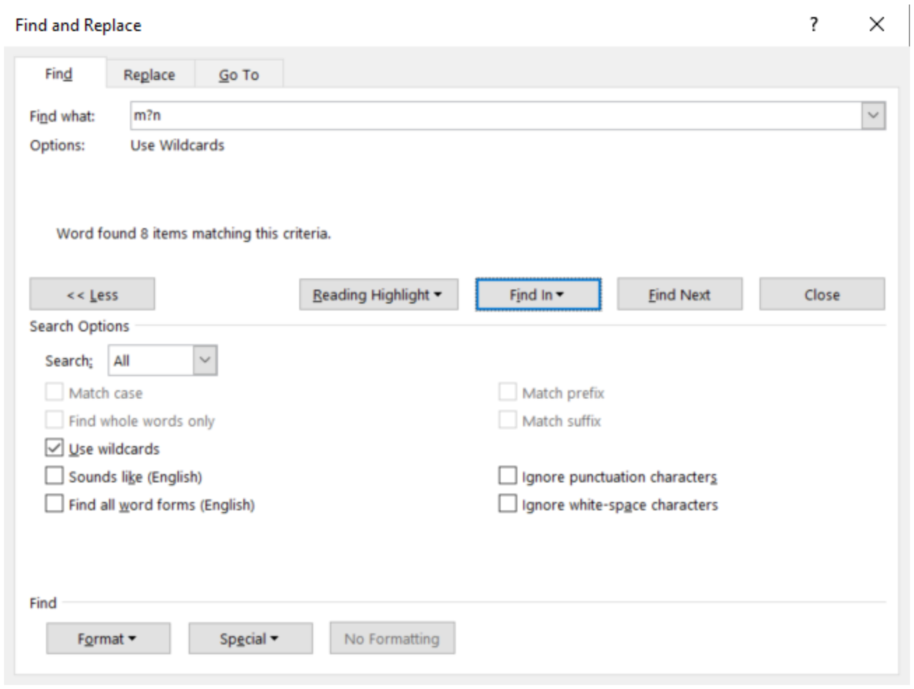
Output from testing was validated against the Microsoft Word search function. Each key word in the trie was searched for in the Word version of the input text and the number of instances identified by Word was ensured to be identical to the number identified by the code. A sample output from running Aho-Corasick on "rainbow.txt" is shown in Figure 5 below:



```
[Key Words]
m*n
source
Saltwater
sm*ller
refract
c*l*r
glass
Rainbow
headphones
jacket

Words Found (iter_long):

Words not found:
headphones
glass
sm*ller
jacket
m*n
c*l*r

[number of mentions]
m*n : 0
source : 2
Saltwater : 1
sm*ller : 0
refract : 11
c*l*r : 0
glass : 0
Rainbow : 85
headphones : 0
jacket : 0

List of all words found (unique):
Rainbow
refract
source
Saltwater
rainbow
```

Figure 5

Wildcard search was tested against the wildcard capability of Microsoft Word by choosing “Advanced Find” from the dropdown menu of the “Find” function on the Home tab. “Use wildcards” was selected from the advanced menu, and the Any Character (?) performed the equivalent function of the asterisk (*) in our implementation.


Find and Replace

Find what: m?n

Options: Use Wildcards

Word found 8 items matching this criteria.

Search Options

Search: All

[✓] Use wildcards

Figure 6

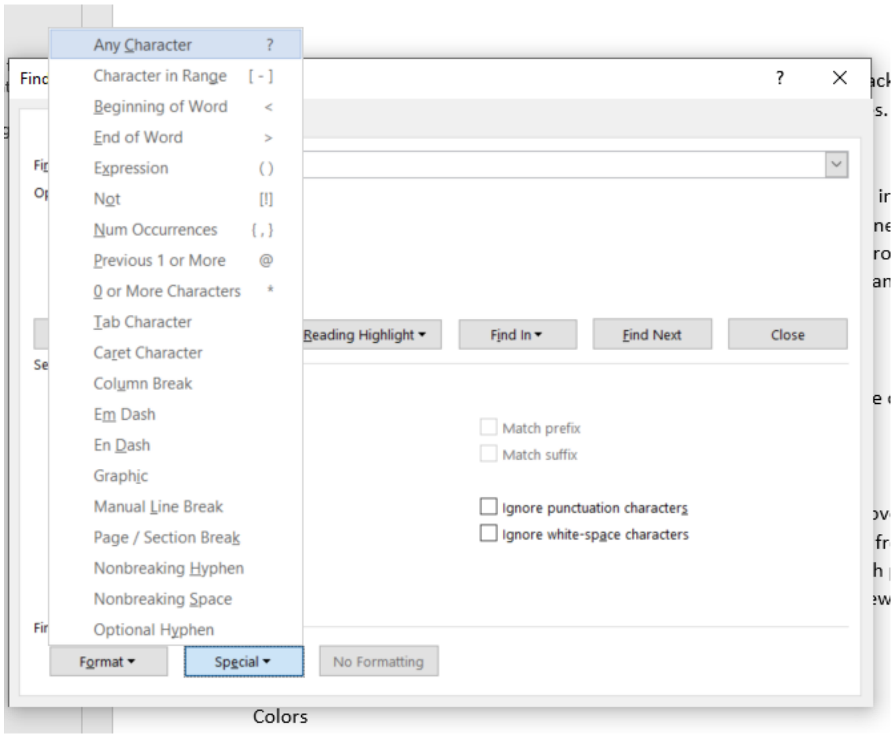


Figure 7

Table 1: Data Summarization

Key Word	Aho-Corasick (case sensitivity = false, wildcards = false)	Aho-Corasick(case sensitivity = true, wildcards = false)	Aho-Corasick (case sensitivity = false, wildcards = true)	Aho-Corasick (case sensitivity = true, wildcards = true)	Microsoft Word (case sensitivity=true, wildcards=true)	Microsoft Word (case sensitivity=false, wildcards=false)
m*n	0	0	9	8	8	0
source	2	2	2	2	2	2
Saltwater	1	1	1	1	1	1
sm*ller	0	0	3	3	3	0
refract	11	11	11	11	11	11
c*l*r	0	0	11	10	10	0
glass	0	0	0	0	0	0
Rainbow	85	11	85	11	11	85
headphones	0	0	0	0	0	0
jacket	0	0	0	0	0	0

As can be seen by the data summarized in Table 1, the algorithm implementation and Microsoft word produced identical results when the same parameters were utilized, verifying not only the basic implementation but also the successful execution of both wildcard searching and case sensitivity. By default, case sensitivity is disabled for Word search, but is automatically enabled when wildcard search is enabled, hence the two columns for Word results.

Some key comparisons to highlight: if case sensitivity is enabled and wildcard search is disabled, the algorithm outputs 11 instead of 85 mentions for the key word "Rainbow," which also matches the instances found in Word when just case sensitivity is enabled. The key word "sm*ller" yields 0 number of mentions despite the word "smaller" being present in the text when wildcards is disabled, but yields 3 when wildcard search is enabled for both the code and in Word. Similar results can be seen for the key word "c*l*r," which is found 10 or 11 times depending on whether or not case sensitivity is enabled while wildcard search is enabled, but found 0 times if wildcard search is disabled. Further breakdown of the results from rainbow.txt are included in the Sample Results section along with the results from searching a much longer piece of text.

Sample Results

We ran our Aho-Corasick implementation on two large texts. The first one is a portion of an article about rainbows taken from the National Geographic website [1]. It consists of 1311 words. The second one is the book Harry Potter and the Sorcerer's Stone by J. K. Rowling [2] consists of 71143 words. We had two configuration options to support case sensitivity and wild cards. Our results include outputs for all combinations of these options.

Table 2: Results for Rainbow.txt

Text size: 1311 words

Keywords: m*n source Saltwater sm*ller refract c*l*r glass Rainbow headphones jacket

Parameters	Match count	Words not found	List of all unique words found	Runtime (sec)
case_sensitive = False, wildcards = False	m*n : 0 source : 2 Saltwater : 1 sm*ller : 0 refract : 11 c*l*r : 0 glass : 0 Rainbow : 85 headphones : 0 jacket : 0	sm*ller headphones jacket c*l*r glass m*n	rainbow source Rainbow antisolar Saltwater refract	0.0055
case_sensitive = True, wildcards = False	m*n : 0 source : 2 Saltwater : 1 sm*ller : 0 refract : 11 c*l*r : 0 glass : 0 Rainbow : 11 headphones : 0 jacket : 0	sm*ller headphones jacket c*l*r glass m*n	source Rainbow antisolar Saltwater refract	0.0079
case_sensitive = False, wildcards = True	m*n : 9 source : 2 Saltwater : 1 sm*ller : 3 refract : 11 c*l*r : 11 glass : 0 Rainbow : 85 headphones : 0 jacket : 0	headphones glass jacket	rainbow source Rainbow min mon color antisolar Color cular Man man Saltwater refract men smaller	0.0090
case_sensitive = True, wildcards = True	m*n : 8 source : 2 Saltwater : 1 sm*ller : 3 refract : 11 c*l*r : 10 glass : 0 Rainbow : 11 headphones : 0 jacket : 0	headphones glass jacket	source Rainbow min mon color antisolar cular man Saltwater refract men smaller	0.0069

Table 2 shows the results for rainbow.txt. We have 10 keywords consisting of wildcards, upper case letters, related and unrelated words. For all configurations, the keywords are successfully found. The search takes 0.0055 seconds with case-sensitivity and wildcards disabled. When both of them are enabled, it takes 0.069 seconds.

Table 3: Results for HP_1.txt

Text size: 71143 words
Keywords: iphone CAR Potter Dudley Privet keyboard scratching algorithm Draconis op*nion m*n V*rn*n freak

Parameters	**Match count**	**Words not found**	**List of all unique words found**	**Runtime (sec)**

case_sensitive = False,
wildcards = False
	iphone : 0
	CAR : 170
	Potter : 134
	Dudley : 141
	Privet : 16
	keyboard : 0
	scratching : 3
	algorithm : 0
	Draconis : 1
	op*nion : 0
	m*n : 0
	V*rn*n : 0
	freak : 1
		keyboard
		iphone
		V*rn*n
		op*nion
		algorithm
		m*n
			Potter
			POTTER
			car
			scratching
			DUDLEY
			Dudley
			CAR
			Draconis
			freak
			Privet
				0.1969

case_sensitive = True,
wildcards = False
	iphone : 0
	CAR : 109
	Potter : 133
	Dudley : 138
	Privet : 15
	keyboard : 0
	scratching : 3
	algorithm : 0
	Draconis : 1
	op*nion : 5
	m*n : 769
	V*rn*n : 116
	freak : 1
		iphone
		algorithm
		keyboard
			Potter Mon MIN
			scratching mkn CAR opinion
			Myn MAN mUn mOn mun Men
			msN Dudley man Man men
			MEN car min
			DUDLEY msn MrN freak
			Privet mAn mEn Vernon
			POTTER mon
			meN Min Draconis mIn myn
				0.2481

case_sensitive = False,
wildcards = True
	iphone : 0
	CAR : 1
	Potter : 132
	Dudley : 140
	Privet : 16
	keyboard : 0
	scratching : 3
	algorithm : 0
	Draconis : 1
	op*nion : 0
	m*n : 0
	V*rn*n : 0
	freak : 1
		keyboard
		iphone
		V*rn*n
		op*nion
		algorithm
		m*n
			Potter
			scratching
			Dudley
			CAR
			Draconis
			freak
			Privet
				0.1931

case_sensitive = True,
wildcards = True
	iphone : 0
	CAR : 1
	Potter : 132
	Dudley : 137
	Privet : 16
	keyboard : 0
	scratching : 3
	algorithm : 0
	Draconis : 1
	op*nion : 5
	m*n : 739
	V*rn*n : 116
	freak : 1
		iphone
		algorithm
		keyboard
			Potter scratching
			mkn CAR opinion
			mUn mOn mun
			Dudley man
			men min msn
			freak Privet
			mAn mEn
			Vernon mon
			Draconis
			mIn myn
				0.1767

Table 3 shows the results for HP_1.txt. Here, we have defined 13 keywords, again, consisting of wildcards, upper case letters, related and unrelated words. The keywords are successfully found with their corresponding match-counts for all configurations. It takes 0.1969 seconds to finish searching with case-sensitivity and wildcards disabled. When both of them are enabled, it takes 0.1767 seconds.

The complexity of the Aho-Corasick algorithm is O(N + L + Z), where Z is the count of matches, text of length N, L is the length of the M patterns. Thus, we can expect the runtimes to increase polynomially as the lengths L and N and the number of matches Z increases.

As we can see from the results, the Aho-Corasick algorithm performs multiple string searching in large texts with very low runtime. It is also able to support case-sensitivity and wildcard searching efficiently as well.

References

Texts

1. Rainbows: https://www.nationalgeographic.org/encyclopedia/rainbow/ (Armstrong)
2. Harry Potter and the Sorcerer's Stone: https://pdfstop.com/category/harry-potter-1/

Code

1. https://github.com/WojciechMula/pyahocorasick
2. https://github.com/abusix/ahocorapy