How to analyze large amounts of data with Rapid Minor

Introduction

Here I wanted to show how the rapid program with text mining can work. I wanted to explain more about how to be able to mine large amounts of data at once in order to find key text that we need to tell a mood, idea, or understanding of a population ideally.

Methods/R Code

First, we uploaded data into the rapid minor program after downloading it. Once the data was downloaded, we then manipulated different operators in order to help let the system know what to do. Next, we had to download two extensions in order to use the operators from those extensions. We downloaded the Aylien text analysis reader as well as the text document processor in order to analyze the text we have. Once this was done we had to connect the read document operator to the open processes window. We then went into it and linked the downloaded text we were going to analyze to it.

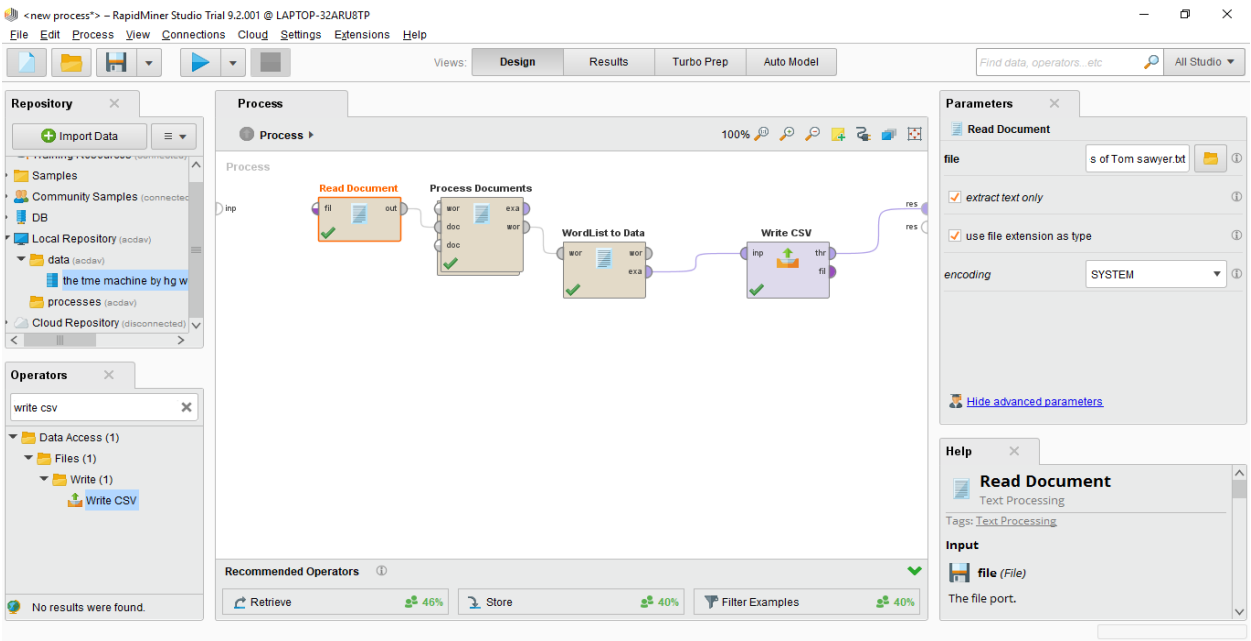
I decided to do The Adventures of Tom Sawyer and the Time Machine by HG Wells in text format. I then loaded their CSV files into the repository in order to be able to access their data and analyze then in the processing stage. Once I uploaded the texts (I did this for both), I then added the process documents operator. Once I added this stage I entered into the actual block and added more operators to specify the process. I added the Tokenize, Transform Cases, and Filter Stopwords (English). Once these were added I went back to the main process and added the Generate n-grams (terms), and then the filter tokens (By content) operators to the entire process sequence on the back of the read document and processing steps.

All these steps added help to transform the cases to upper and lower case when the machine goes through the document. Also, it helps to filter out words with minimal meaning (and, a, the, etc.), and to

split out the words in the text and remove the punctuation. In the main process we added the latter two processes to help generate how many times a word may have occurred as well as help decipher what the generated words would be separated with. In the main process pane, we then added a write to word list data step in order to be able to write to ta file. We then added a read-out process in order to export the final file to a document on the computer (CSV, etc.).

Results/Outputs

Below are the different steps of the process along with thetext imported and the final output.



Above is the tom sawyer file uploaded on the right in the read document tab for parameters.

Process (2 results. Process results)
Completed: Apr 6, 2019 12:17:12 PM (execution time: 0 s)

ExampleSet (Process Documents)
Result not stored in repository.

Data Table
Number of examples = 1
250 attributes:

Role	Name	Type	Range	Missings	Comment
-	abandon_firewood	real	= [?...?]; mean =?	no missing values	-
-	abandoned_ruins	real	= [?...?]; mean =?	no missing values	-
-	abide_terms	real	= [?...?]; mean =?	no missing values	-
-	able_people	real	= [?...?]; mean =?	no missing values	-

Word List (Process Documents)
Result not stored in repository.

```
Contains 14331 entries.
This list has been created on 1 documents assigned to 0 labels
```

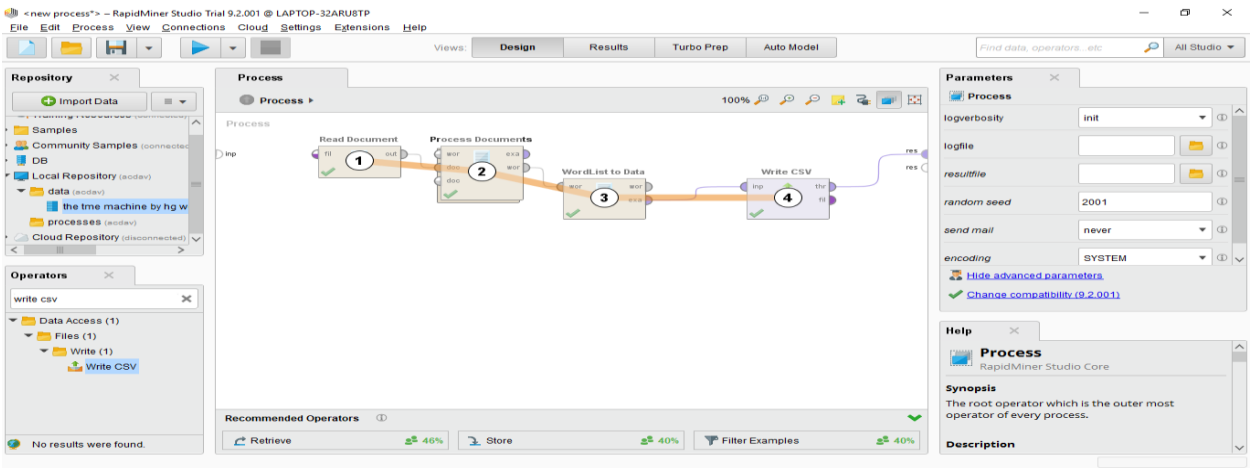
Process (1 results. Process results)
Completed: Apr 6, 2019 1:34:10 PM (execution time: 1 s)

ExampleSet (WordList to Data)
Result not stored in repository.

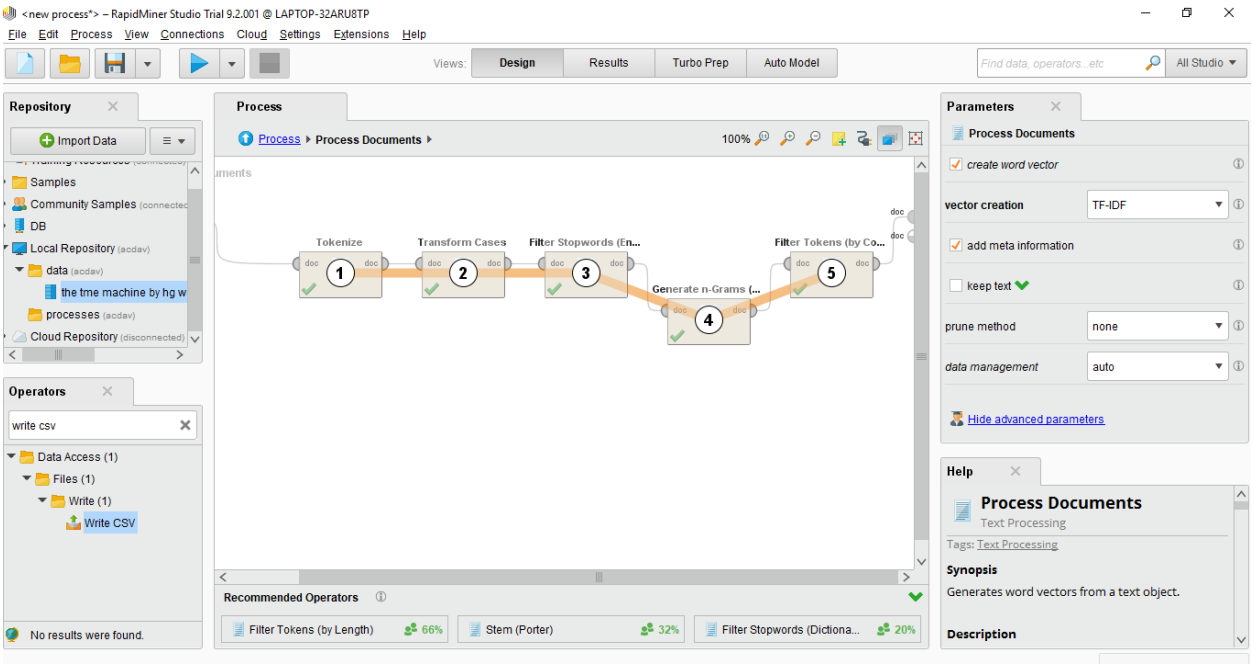
Data Table
Number of examples = 29386
3 attributes:

Role	Name	Type	Range	Missings	Comment
-	word	polynominal	= {abandoned_cobwebs, abandoned_negro, abandoned_phrenologist, abandoned_slaughter, abash_boy, abashed_uncomfortable, abide_joe, abide_terms, ablaze_bliss, ...}	no missing values	-
-	in documents	integer	= [?...?]; mean =?	no missing values	-
-	total	integer	= [?...?]; mean =?	no missing values	-

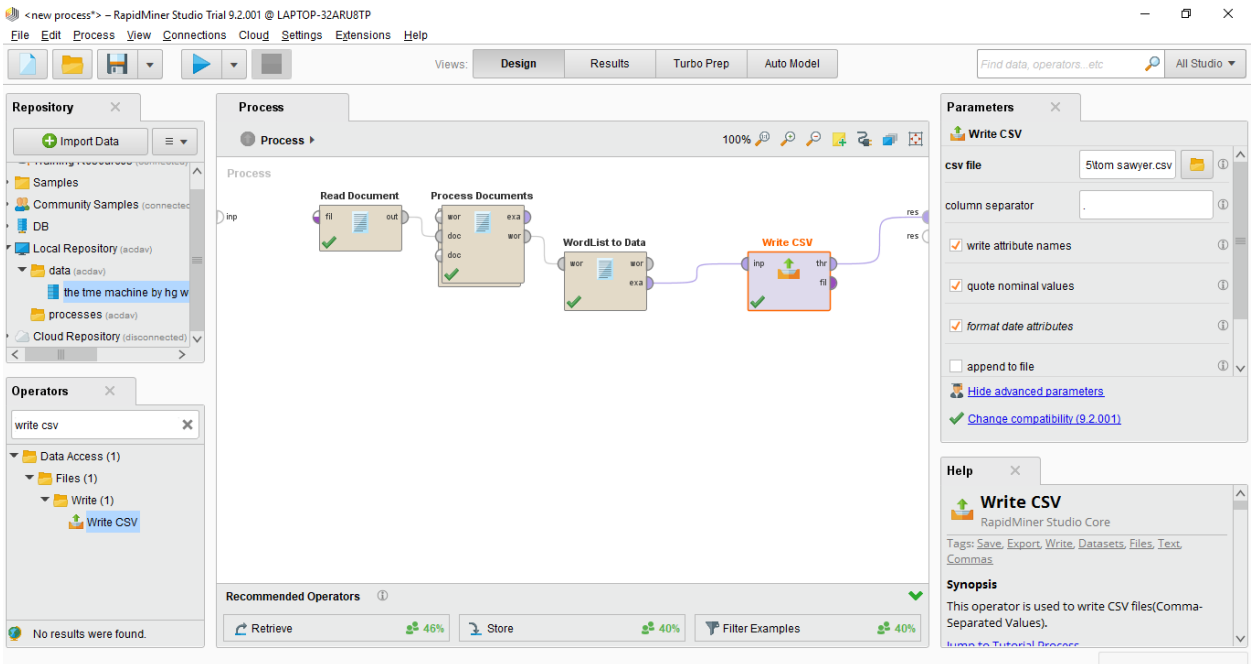
Above are the two reports generated when the book was uploaded showing the amount of sentences and the other the amount of characters/words.



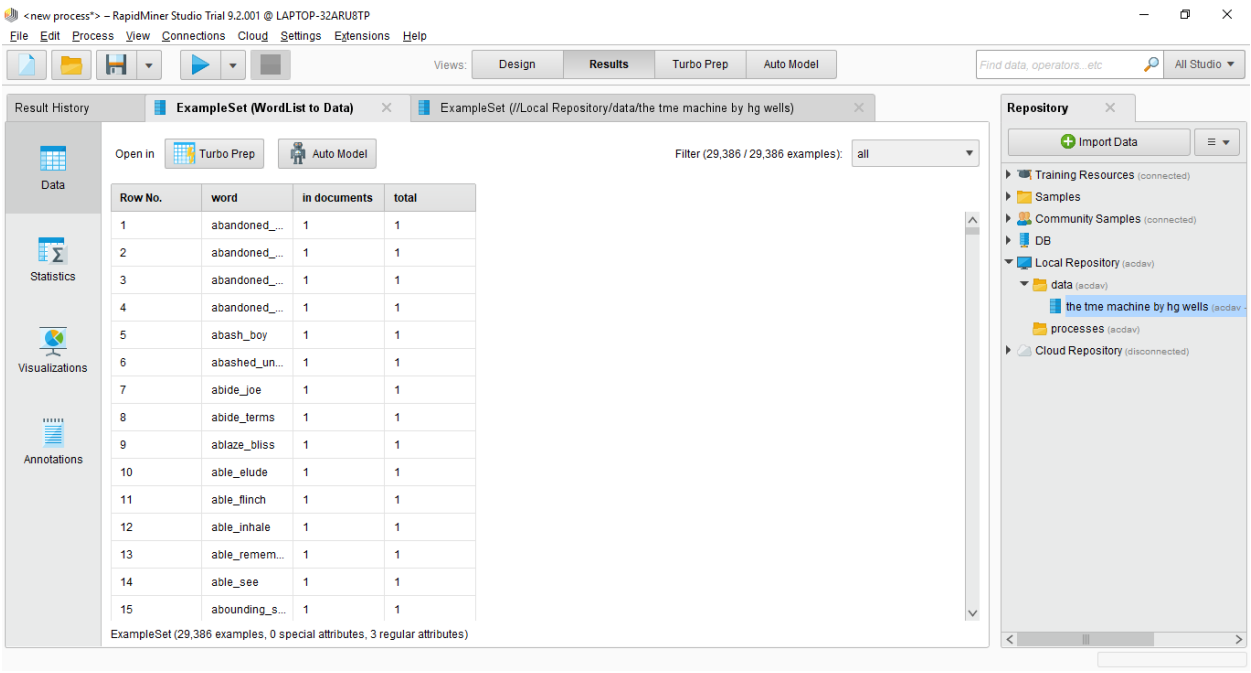
Above is the main process in the sequence in which it was added when building the process.



Above is the sequence as well as the operators added within the process document phase of the overall process.



Above is the write CSV file that was created at the end of the process.



Above is the final output generated by the write to CSV and overall process.

Analysis of Results

We found that overall there were a lot of words in the Tom Sawyer novel. More than the HG wells novel (which had 4,939 sentences). So, I found that this process was very efficient and fast with certain types of data. Rapid minor did have a great feel to it while working. Also, I was not able to generate a dictionary to use from Python and uploaded to make the analysis work better like in the example. However, this did allow me to see the process from rapid minor and I thought it was still a great experience overall.

Conclusion

Here I wanted to display the usefulness of Rapid Minor and how it can be very beneficial in the mining of large text documents in order to separate out certain words, phrases, and key finding that one is looking for. There will be times in which large data will need to be rapidly skimmed in the search for important information and one can easily use a software like rapid minor to accomplish that. This program could

allow an individual to rapidly find what they are looking for as well as help others find what they need within large seas of information. I am sure other programs like Python and R would be more robust in certain areas of this practice, but it is always good to see other programs at work on such an easy to operate interface that involved minimal code writing.

Resources:

1. Rapid Minor. https://my.rapidminer.com/nexus/account/index.html#downloads
2. Gutenberg text. https://www.gutenberg.org/files/74/74-0.txt
3. Aylien Text Analysis. https://developer.aylien.com/