

NAME: KALYANI AJAY DAWALKAR

CLASS: BE-A                      ROLL NO:11

## EXPERIMENT-04

### Title

Consider a suitable text dataset. Remove stop words, apply stemming and feature selection techniques to represent documents as vectors. Classify documents and evaluate precision, recall.

### Objective:

1. Learn how to tokenize and filter a document into its different words and then do words count for each word in a text document.
2. Learn Text processing through Rapidminer.

### Problem Statement:

Demonstrate Text processing like Remove stop words, apply stemming and feature selection technique to represent document as vector & Classify the documents.

### Outcomes:

1. Students will be able to demonstrate Text processing using Rapidminer
2. Students will be able to see a word list containing all the different words in your document and their occurrence count next to it in the "Total Occurrences" column

**Hardware Requirement:** Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more

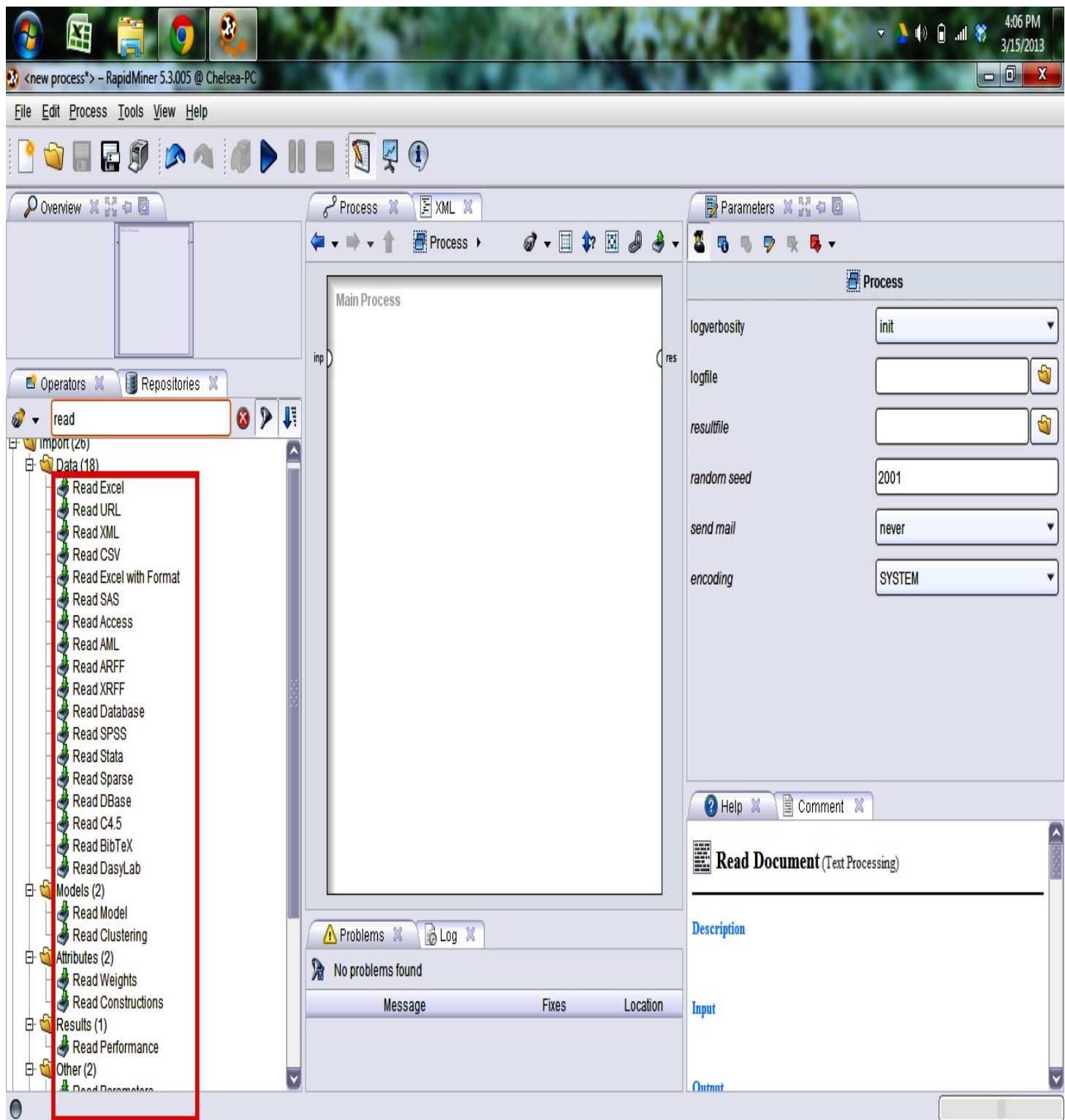
**Software Requirements:** 32/64 bit Linux/Windows Operating System, latest Rapidminer Tool

### Theory:

#### Text Processing Tutorial with RapidMiner

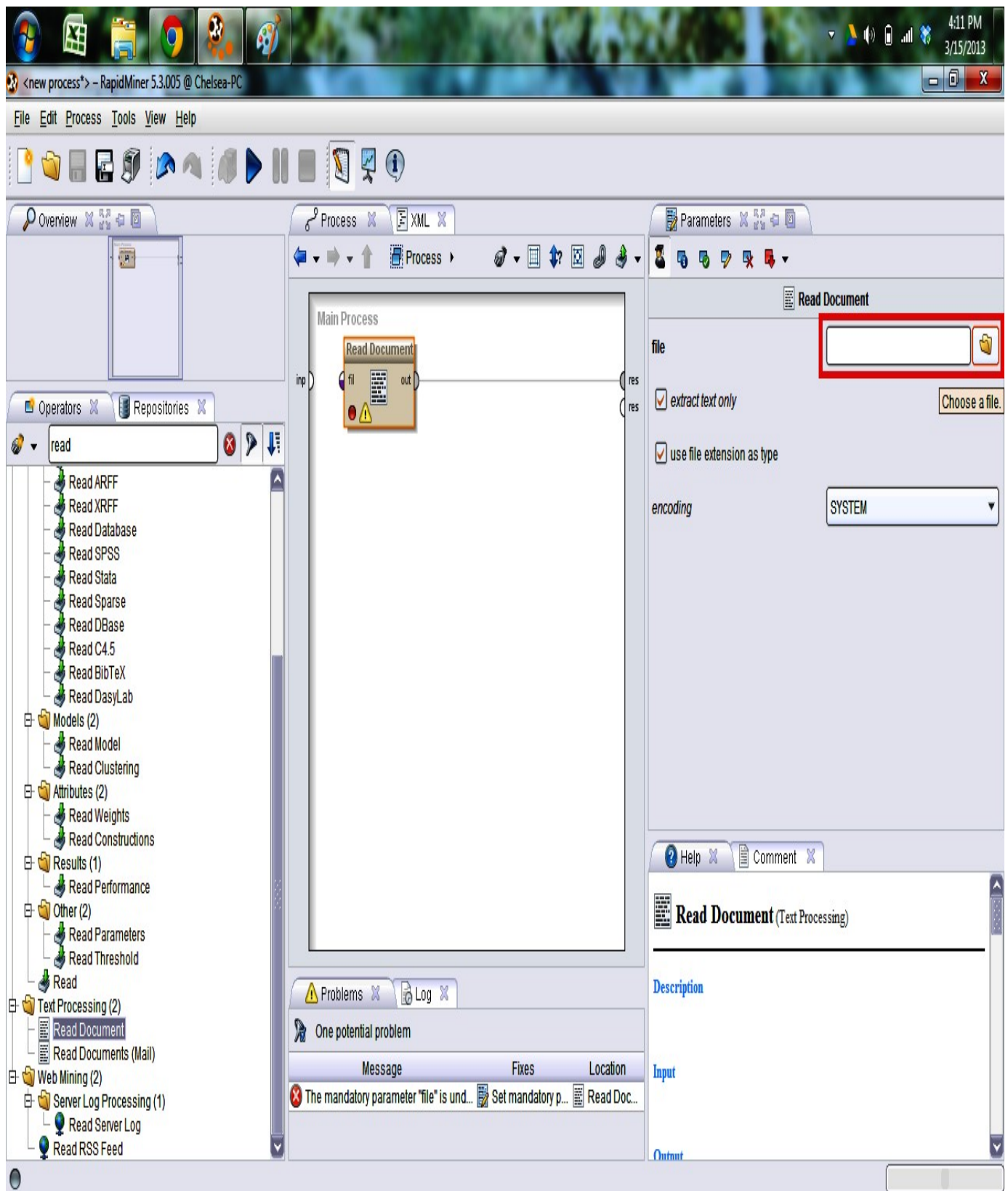
In this Manual, we are going to learn how to tokenize and filter a document into its different words and then do words count for each word in a text document

Open RapidMiner and click "New Process". On the left hand pane of your screen, there should be a tab that says "Operators"- this is where you can search and find all of the operators for RapidMiner and its extensions. By searching the Operators tab for "read", you should get an output like this (you can double click on the images below to enlarge them):



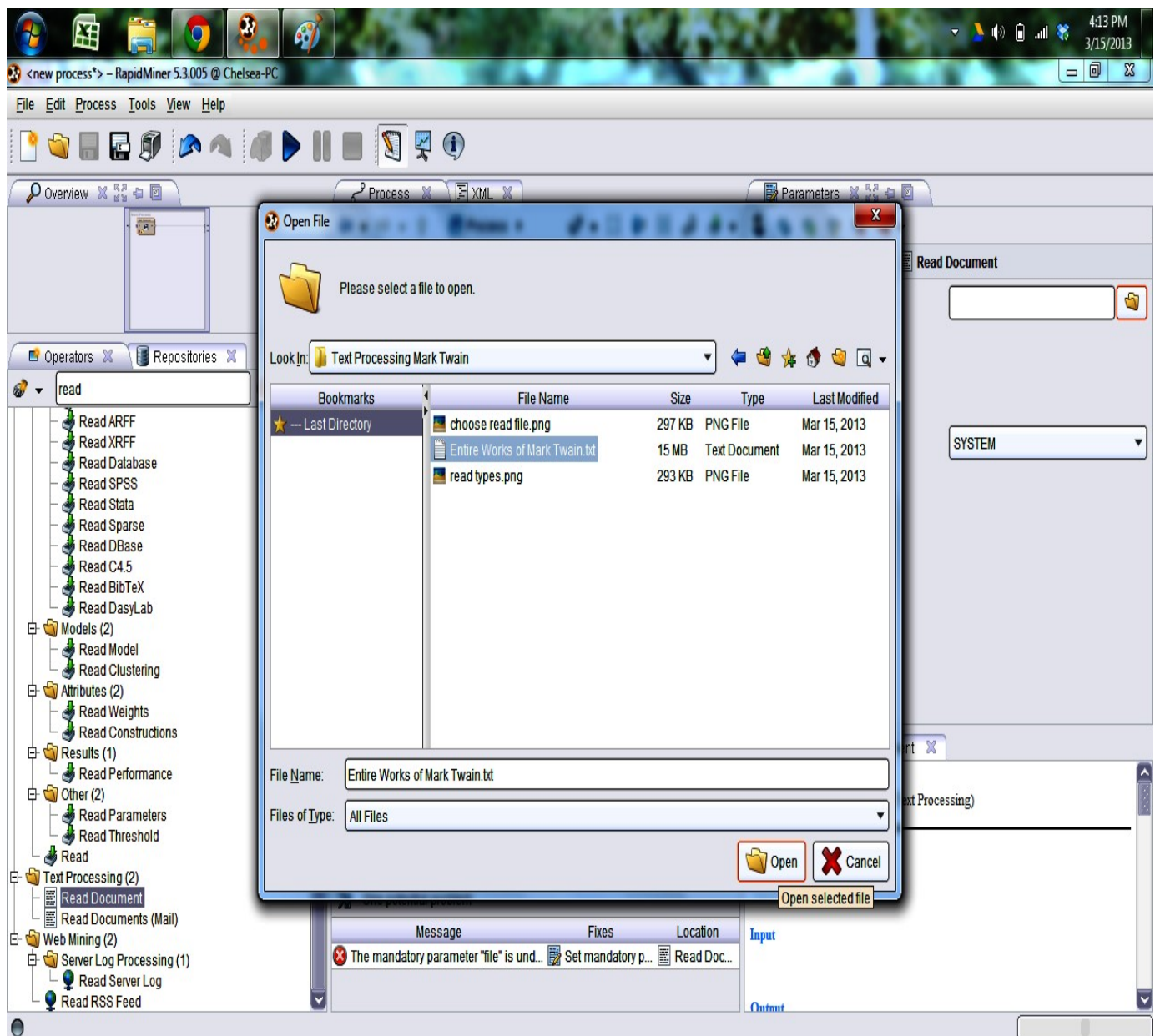
**Figure 1:** Searching the Operators tab for "read"

There are multiple read operators depending on which file you have, and most of them work the same way. If you scroll down, there is a "Read Documents" operator. Select this operator and enter it into your Main Process window by dragging it. When you select the Read Documents operator in the Main Process window, you should see a file uploader in the right-hand pane.



**Figure 2:** Drag and Drop “Read Documents” operator

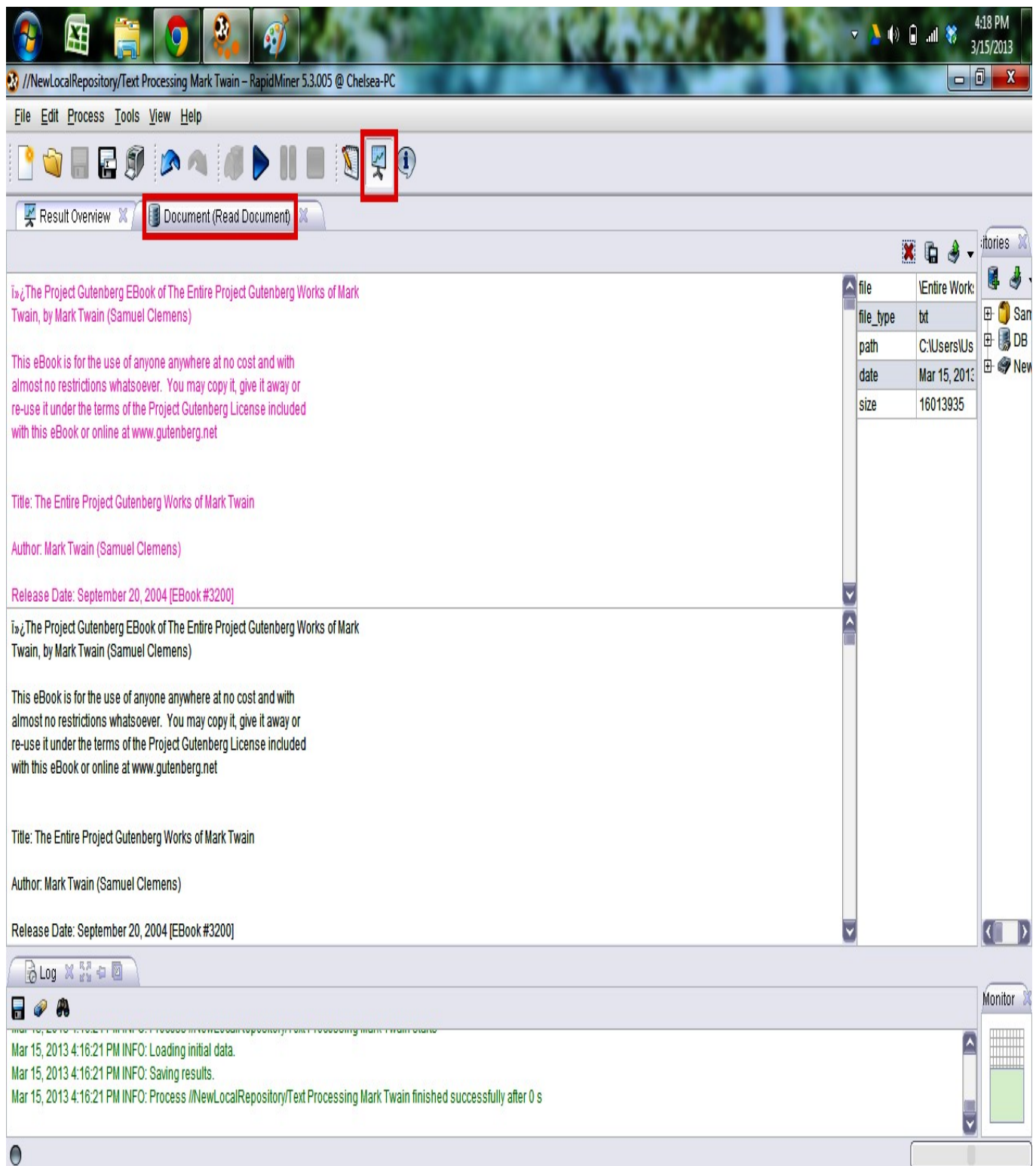
Select the text file you want to use.



**Figure 3:** Select the text file you want to use

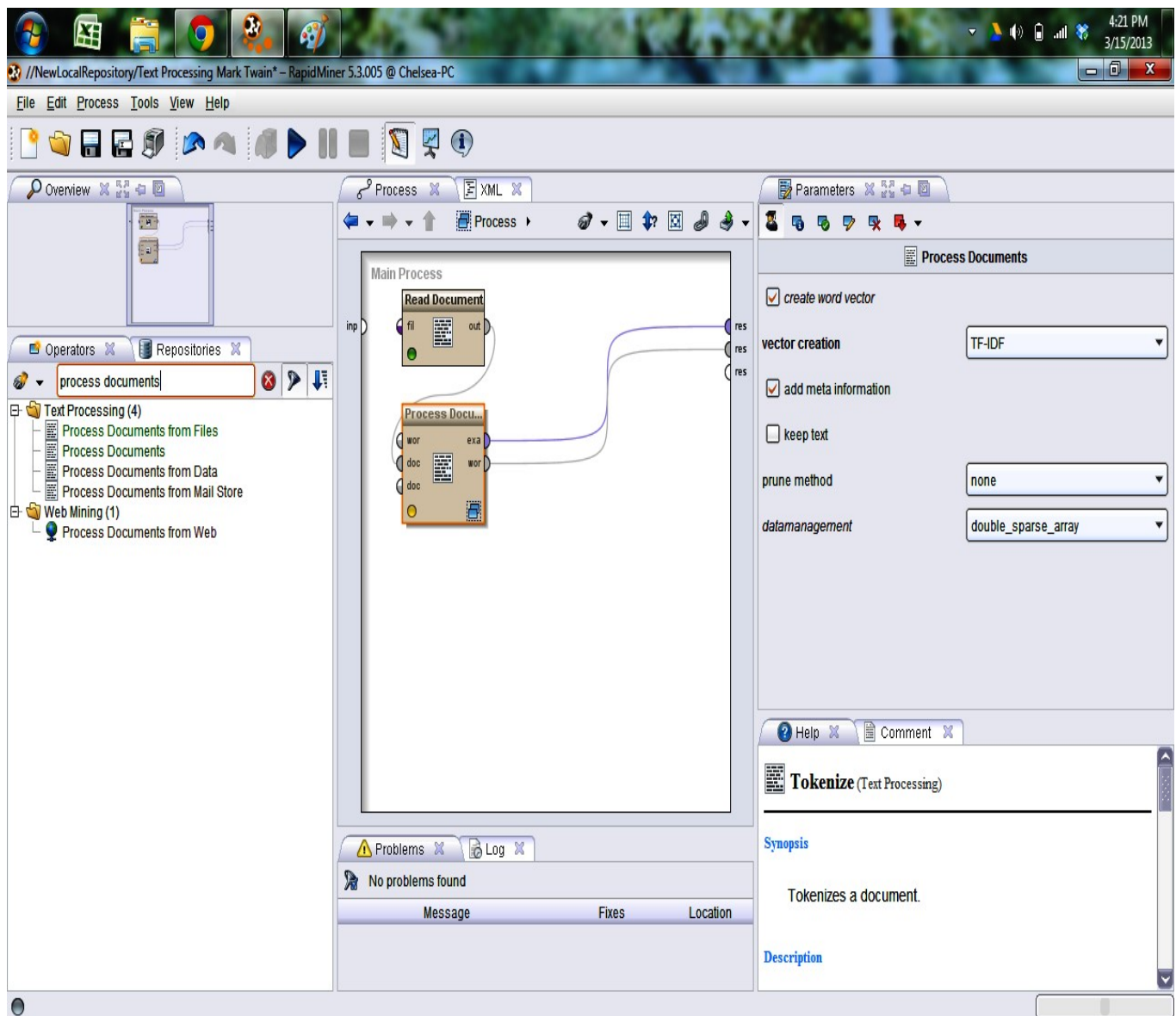
After you have chosen your file, make sure that the output port on the Read Documents operator is connected to the "res" node in your Main Process. Click the "play" button to check that your file has been received correctly. Switch to the results perspective by clicking the icon that looks like a display chart above the "Process" tab at the top of the Main Process pane. Click the "Document (Read Document)" tab. Your output text should look something like this depending on the file you have chosen to process:





**Figure 4: Run the Process**

Now we will move on to processing the document to get a list of its different words and their individual count. Search the Operators list for "Process Documents". Drag this operator the same way as you did for the "Read Documents" operator into the main panel.



**Figure 5:** Search the Operators list for "Process Documents"

Double click the Process Documents operator to get inside the operator. This is where we will link operators together to take the entire text document and split it down into its word components. This consists of several operators that can be chosen by going into the Operator pane and looking at the Text Processing folder. You should see several more folders such as "Tokenization", "Extraction", "Filtering", "Stemming", "Transformation", and "Utility". These are some of the descriptions of what you can do to your document. The first thing that you would want to do to your document is to tokenize it. Tokenization creates a "bag of words" that are contained in your document. This allows you to do further filtering on your document. Search for the "Tokenize" operator and drag it into the "Process Documents" process.

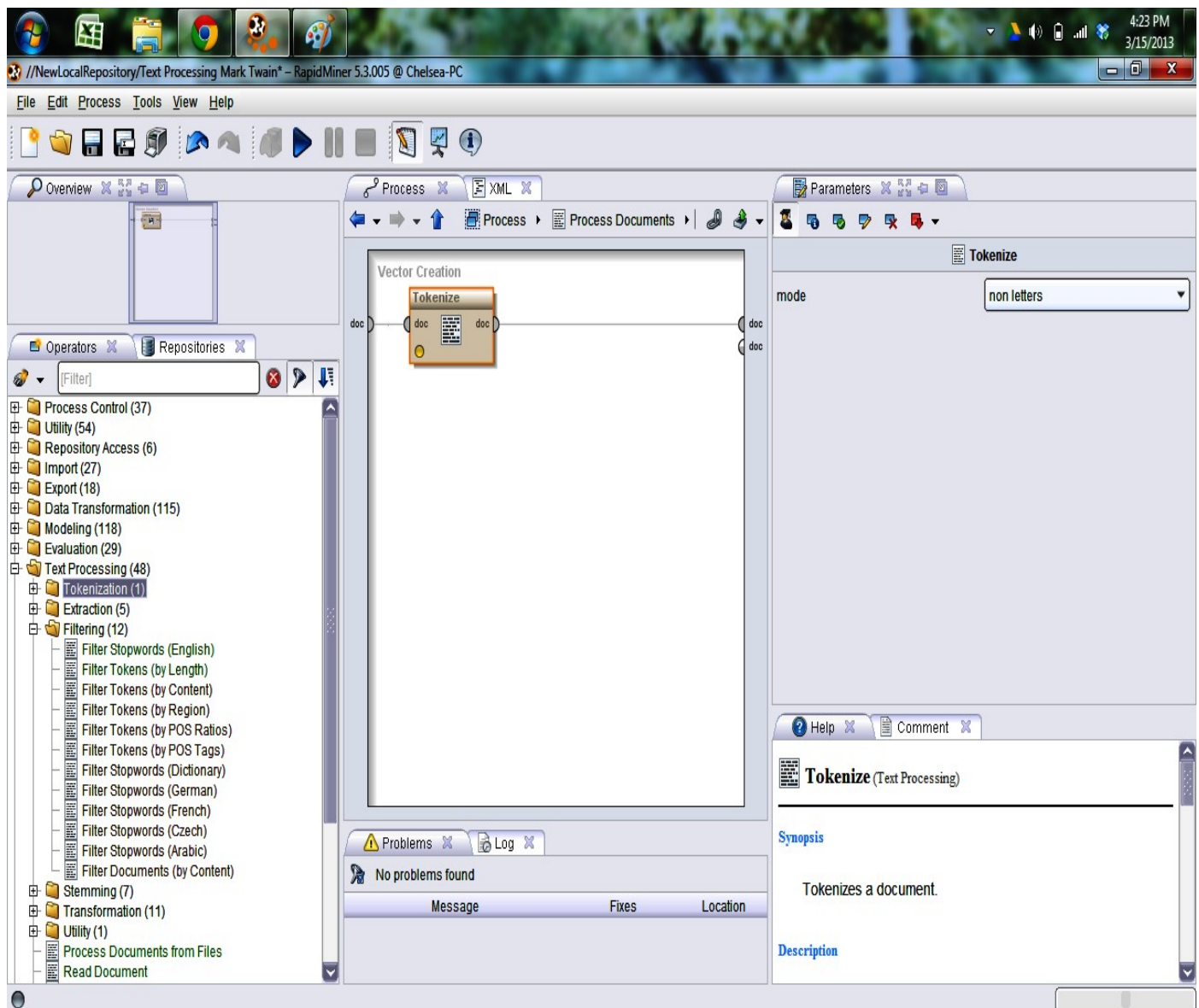
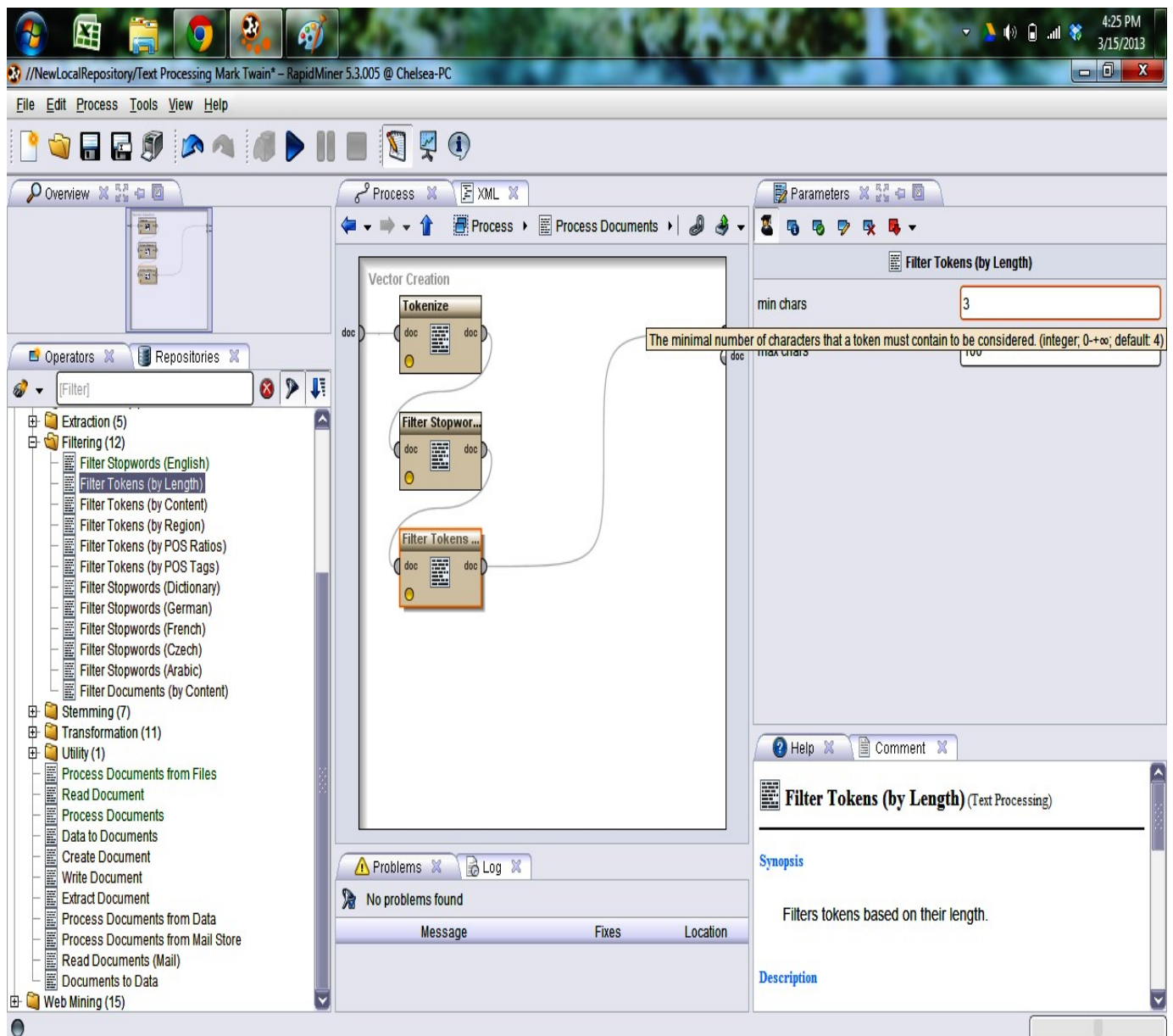


Figure 6 Search for the "Tokenize" operator

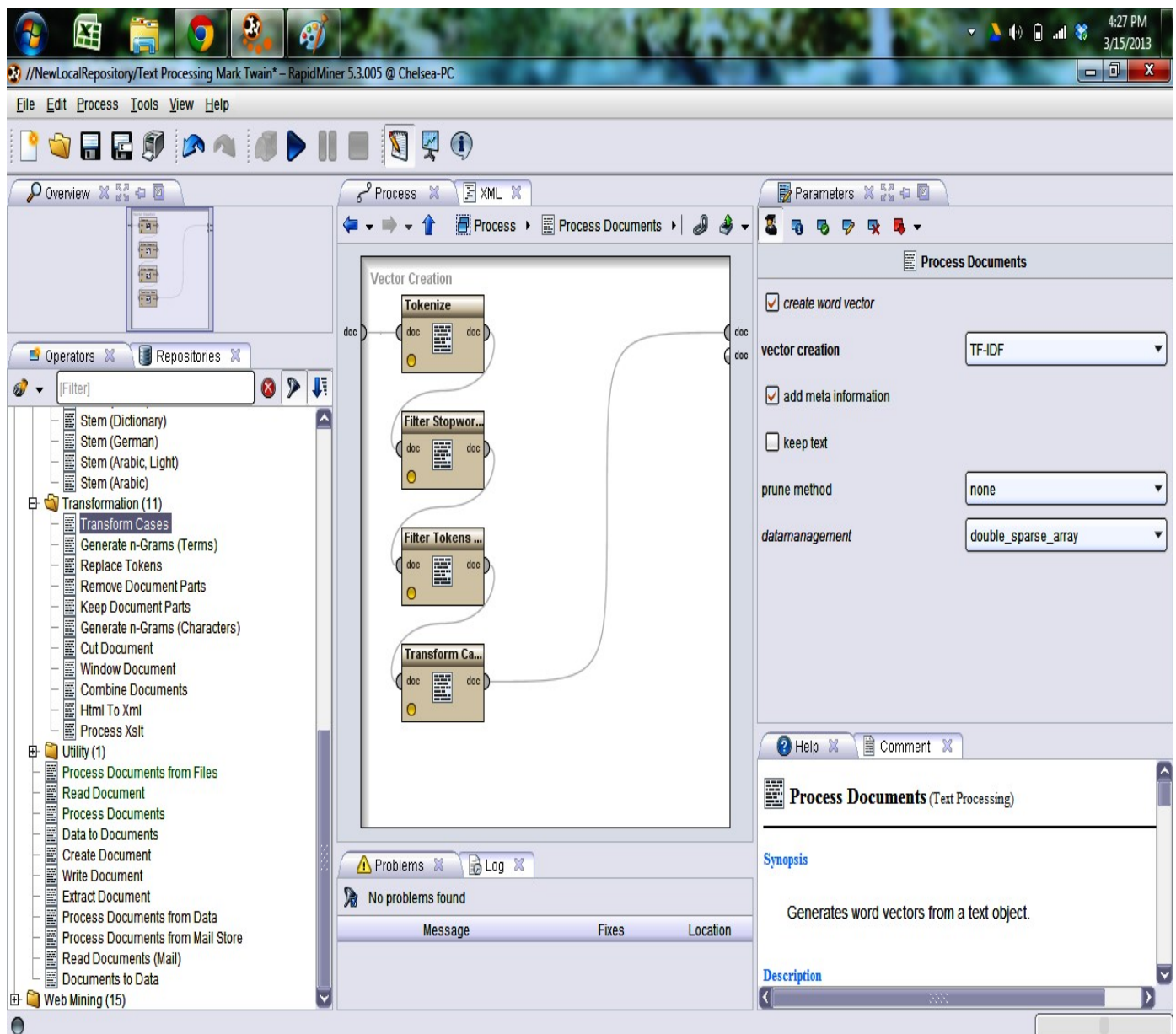
Connect the "doc" node of the process to the "doc" input node of the operator if it has not automatically connected already. Now we are ready to filter the bag of words. In "Filtering" folder under the "Text Processing" operator folder, you can see the various filtering methods that you can apply to your process. For this example, I want to filter certain words out of my document that don't really have any meaning to the document itself (such as the words a, and, the, as, of, etc.); therefore, I will drag the "Filter Stopwords (English)" into my process because my document is in English. Also, I want to filter out any remaining words that are less than three characters. Select "Filter Tokens by Length" and set your parameters as desired (in this case, I want my min number of characters to be 3, and my max number of characters to be an arbitrarily large number since I don't care about an upper bound). Connect the nodes of each subsequent operator accordingly as in the picture.



**Figure 7** Select the operator "Transform Cases" and drag it into the process.

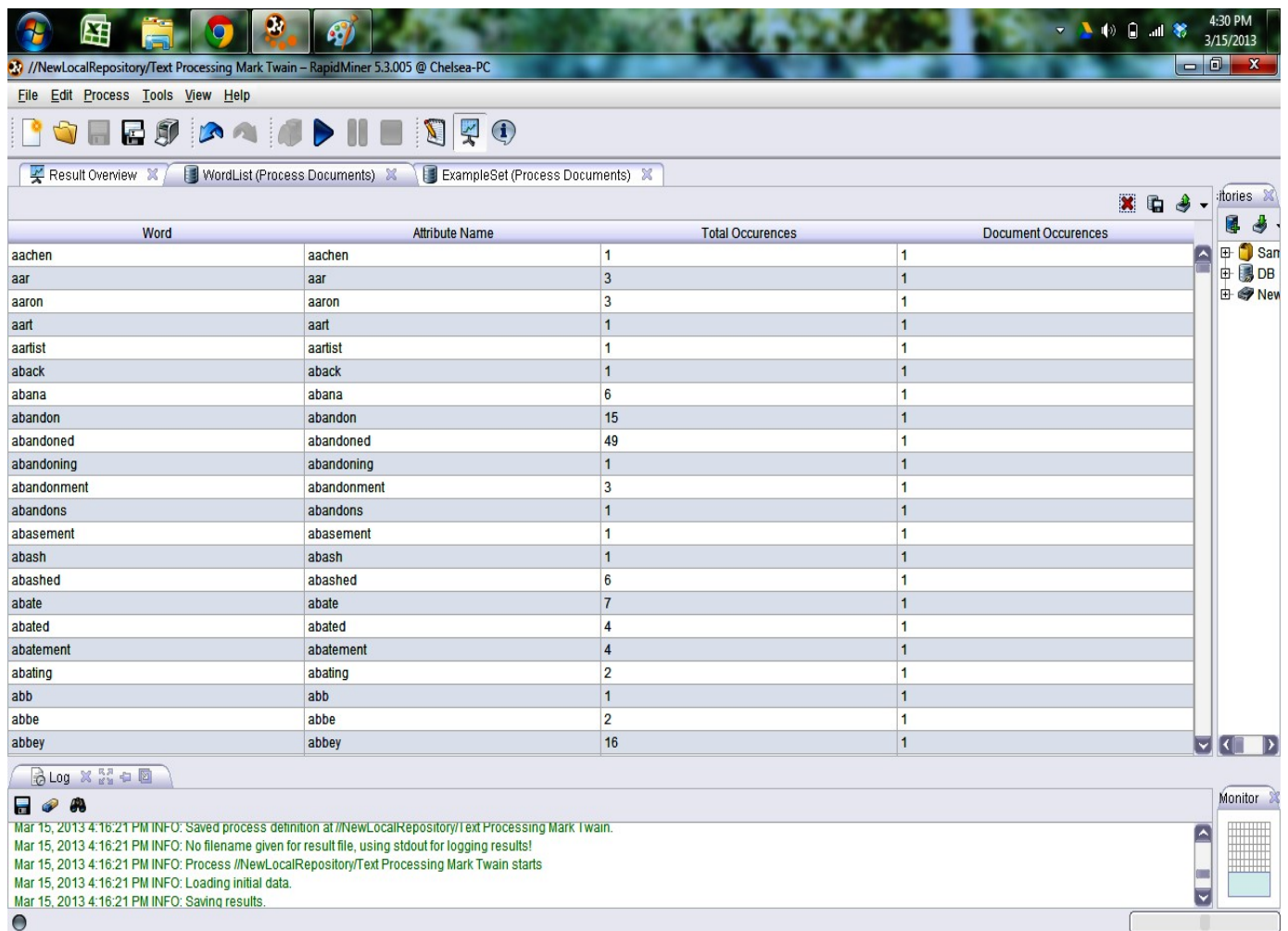
After I filtered the bag of words by stopwords and length, I want to transform all of my words to lowercase since the same word would be counted differently if it was in uppercase vs. lowercase. Select the operator "Transform Cases" and drag it into the process.





**Figure 8** Checks all nodes connections and clicks the "Play" button to run process

Now that I have the sufficient operators in my process for this example, I check all of my node connections and click the "Play" button to run my process. If all goes well, your output should look like this in the results view:



Word	Attribute Name	Total Occurrences	Document Occurrences
aachen	aachen	1	1
aar	aar	3	1
aaron	aaron	3	1
aart	aart	1	1
aartist	aartist	1	1
aback	aback	1	1
abana	abana	6	1
abandon	abandon	15	1
abandoned	abandoned	49	1
abandoning	abandoning	1	1
abandonment	abandonment	3	1
abandons	abandons	1	1
abase	abase	1	1
abash	abash	1	1
abashed	abashed	6	1
abate	abate	7	1
abated	abated	4	1
abatement	abatement	4	1
abating	abating	2	1
abb	abb	1	1
abbe	abbe	2	1
abbey	abbey	16	1

Log

```

Mar 15, 2013 4:16:21 PM INFO: Saved process definition at //NewLocalRepository/Text Processing Mark Twain.
Mar 15, 2013 4:16:21 PM INFO: No filename given for result file, using stdout for logging results!
Mar 15, 2013 4:16:21 PM INFO: Process //NewLocalRepository/Text Processing Mark Twain starts
Mar 15, 2013 4:16:21 PM INFO: Loading initial data.
Mar 15, 2013 4:16:21 PM INFO: Saving results.

```

**Figure 9** Output should look like this in the results view

## Conclusion

We are now able to see a word list containing all the different words in your document and their occurrence count next to it in the "Total Occurrences" column. If you do not get this output, make sure that all of your nodes are connected correctly and also to the right *type*. Some errors are because your output at one node does not match the type expected at the input of the next node of an operator

---