CHAPTER TWELVE: TEXT MINING

CONTEXT AND PERSPECTIVE

Gillian is a historian and archivist at a national museum in the Unites States. She has recently curated an exhibit on the Federalist Papers. The Federalist Papers are a series of dozens of essays that were written and published in the late 1700's. The essays were published in two different newspapers in the state of New York over the course of about one year, and they were released anonymously under the author name 'Publius'. Their intent was to educate the American people about the new nation's proposed constitution, and to advocate in favor of its ratification. No one really knew at the time if 'Publius' was one individual or many, but several individuals familiar with the authors and framers of the constitution had spotted some patterns in vocabulary and sentence structure that seemed familiar to sections of the U. S. constitution. Years later, after Alexander Hamilton died in the year 1804, some notes were discovered that revealed that he (Hamilton), James Madison and John Jay had been the authors of the papers. The notes indicated specific authors for some papers, but not for others. Specifically, John Jay was revealed to be the author for papers 3, 4 and 5; Madison for paper 14; and Hamilton for paper 17. Paper 18 had no author named, but there was evidence that Hamilton and Madison worked on that one together.

LEARNING OBJECTIVES

After completing the reading and exercises in this chapter, you should be able to:

- Explain what text mining is, how it is used and the benefits of using it.
- Recognize the various formats that text can be in, in order to perform text mining.
- Connect to and import text as a data source for a text mining model.
- Develop a text mining model in RapidMiner including common text-parsing operators such as **tokenization**, **stop word** filtering, **n-gram** construction, **stemming**, etc.

- Apply other data mining models to text mining results in order to predict or classify based on textual analysis.

ORGANIZATIONAL UNDERSTANDING

Gillian would like to analyze paper 18's content in the context of the other papers with known authors, to see if she can generate some evidence that the suspected collaboration between Hamilton and Madison is in fact a likely scenario. She feels like **text mining** might be a good method to analyze the text in a structured way, and has enlisted our help. Having studied all of the Federalist Papers and other writings by the three statesmen who wrote them, Gillian feels confident that paper 18 is a collaboration that John Jay *did not* contribute to—his vocabulary and grammatical structure was quite different from those of Hamilton and Madison, even when all three wrote on the same topic, as they had with the Federalist Papers. She would like to look for word and phrase choice frequencies and present the outcome as part of her exhibit on the papers. We will help Gillian by constructing a text mining model using the text from the Federalist Papers and some standard text mining methodologies.

DATA UNDERSTANDING

Gillian's data set is simple: we will include the full text of Federalist Papers number 5 (Jay), 14 (Madison), 17 (Hamilton), and 18 (suspected collaboration between Madison and Hamilton). The Federalist Papers are available through a number of sources: they have been re-published in book form, they are available on a number of different web sites, and their text is archived in many libraries throughout the world. For this chapter's exercise, the text of these four papers has been added to the book's companion web site. There are four files for you to download:

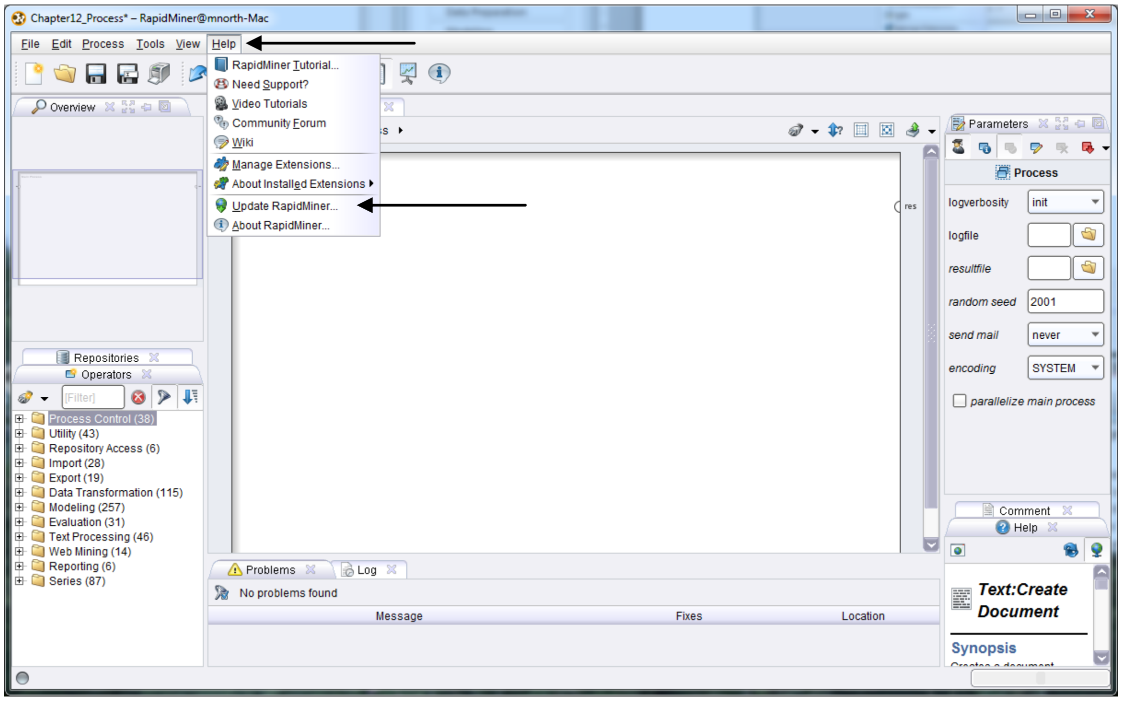
- Chapter12_Federalist05_Jay.txt
- Chapter12_Federalist14_Madison.txt
- Chapter12_Federalist17_Hamilton.txt
- Chapter12_Federalist18_Collaboration.txt.

Please download these now, but do not import them into a RapidMiner repository. The process of handling textual data in RapidMiner is a bit different than what we have done in past chapters. With these four papers' text available to us, we can move directly into the CRISP-DM phase of…

DATA PREPARATION

The text mining module of RapidMiner is an optional add-in. When you installed RapidMiner (way back in Step 4 of the 'Preparing RapidMiner…' section of Chapter 3), we mentioned that you might want to include the Text Processing component. Whether you did or did not at that time, we will need it for this chapter's example, so we can add it now. Even if you did add it earlier, it might be a good idea to complete all of the steps below to ensure your Text Processing add-in is up-to-date.

1) Open RapidMiner to a new, blank process. From the application menu, select Help > Update RapidMiner…



logverbosity	init
logfile	
resultfile	
random seed	2001
send mail	never
encoding	SYSTEM

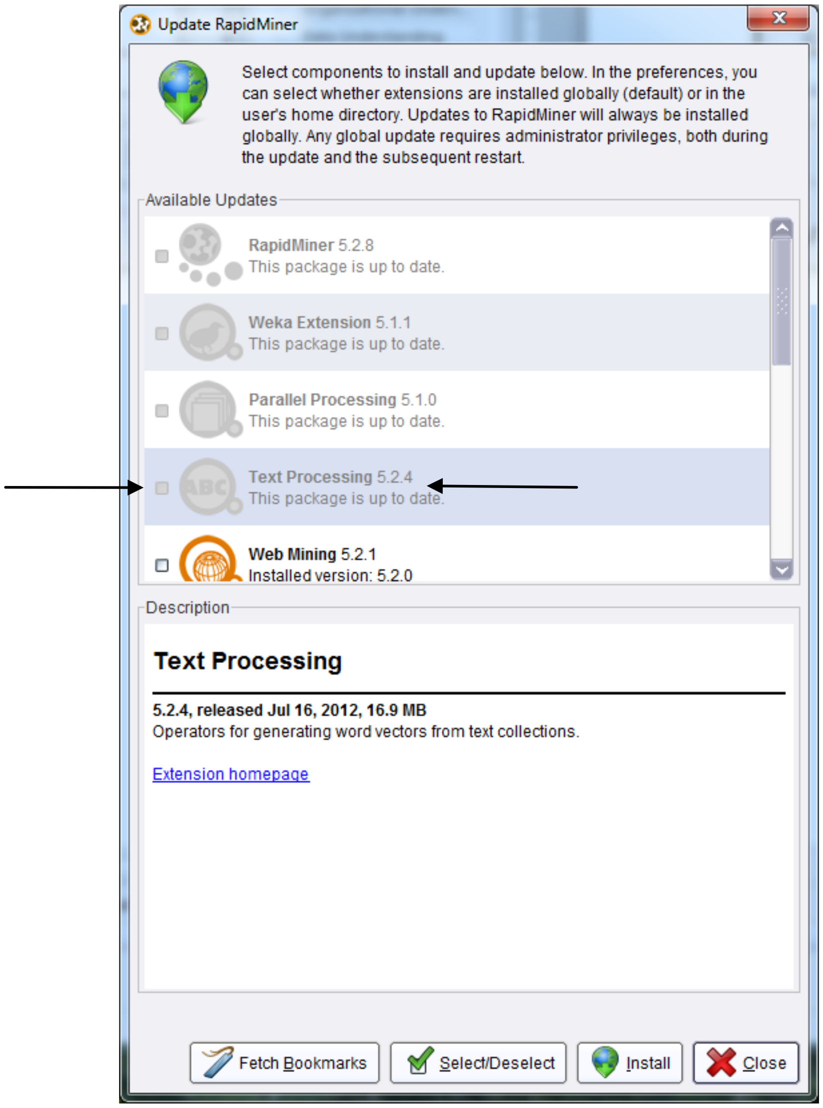
parallelize main process

No problems found

Figure 12-1: Updating RapidMiner add-ins.

2) Your computer will need to be connected to the Internet, so that it can check Rapid-I's servers to see if any updates are available. Once the connection has been established and the software has checked for available updates, you will see a window similar to Figure 12-2. Locate Text Processing in the list (it should be about fourth from the top). If it is grayed out, that means that the add-in is installed and up-to-date on your computer. If it is not installed, or not up to the current version, it will be orange. You can double click the

small square to the left of the Text Processing icon (the circle with 'ABC' in it). Then click the Install button to add or update the module. When it is finished, the window will disappear and you will be back to your main RapidMiner window.


Update RapidMiner

Select components to install and update below. In the preferences, you can select whether extensions are installed globally (default) or in the user's home directory. Updates to RapidMiner will always be installed globally. Any global update requires administrator privileges, both during the update and the subsequent restart.

Available Updates

RapidMiner 5.2.8
This package is up to date.

Weka Extension 5.1.1
This package is up to date.

Parallel Processing 5.1.0
This package is up to date.

Text Processing 5.2.4
This package is up to date.

Web Mining 5.2.1
Installed version: 5.2.0

Description

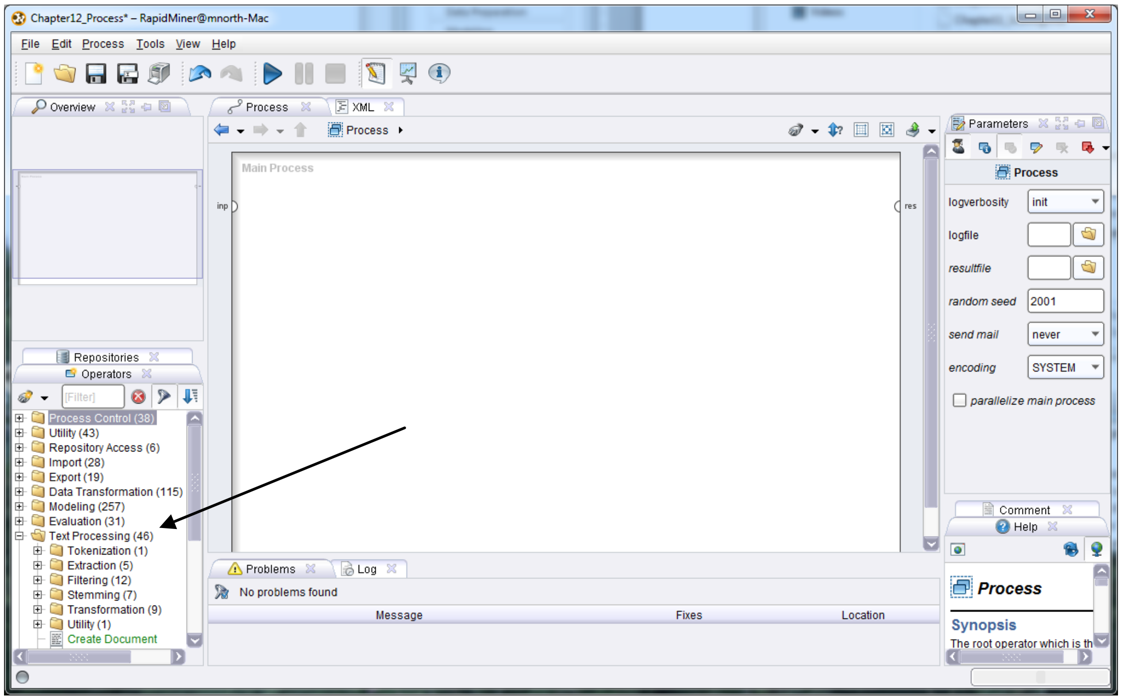
Text Processing

5.2.4, released Jul 16, 2012, 16.9 MB
Operators for generating word vectors from text collections.

Extension homepage

Figure 12-2. Adding/updating the RapidMiner Text Processing add-in.

3) In the Operators tab in the lower left hand area of your RapidMiner window, locate and expand the Text Processing operators folder by clicking on the + sign next to it.

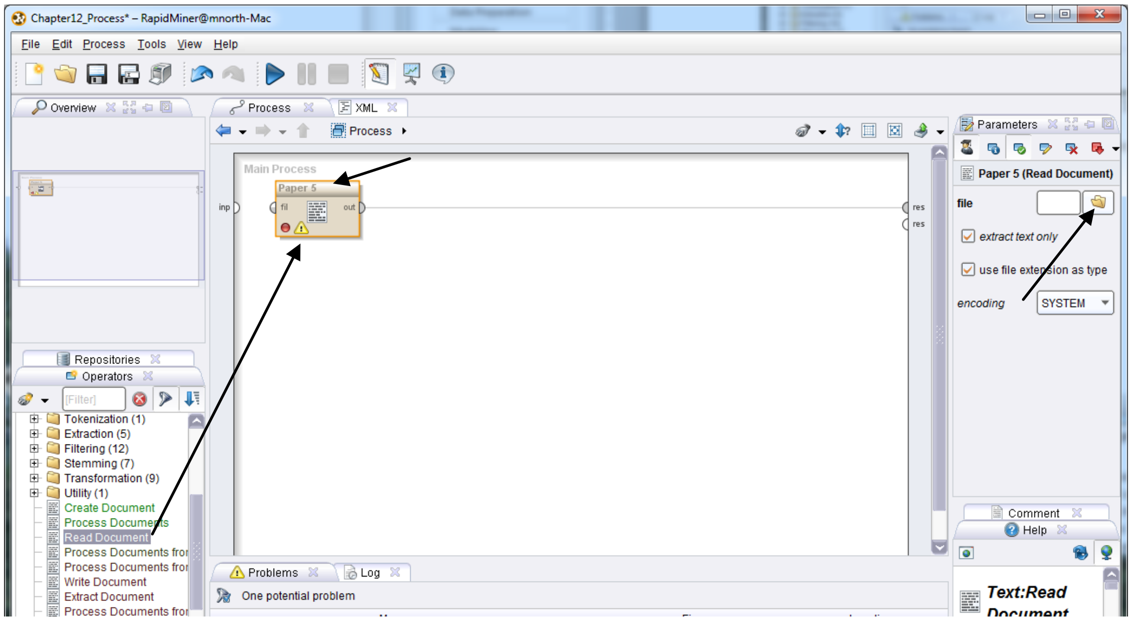


logverbosity: init
logfile:
resultfile:
random seed: 2001
send mail: never
encoding: SYSTEM
parallelize main process: ☐

No problems found

Figure 12-3. Finding tools in the Text Processing operator area.

4) Within the Text Processing menu tree, there is an operator called Read Document. Drag this operator and drop it into your main process window. Right click on it and rename it 'Paper 5', as shown in Figure 12-4.

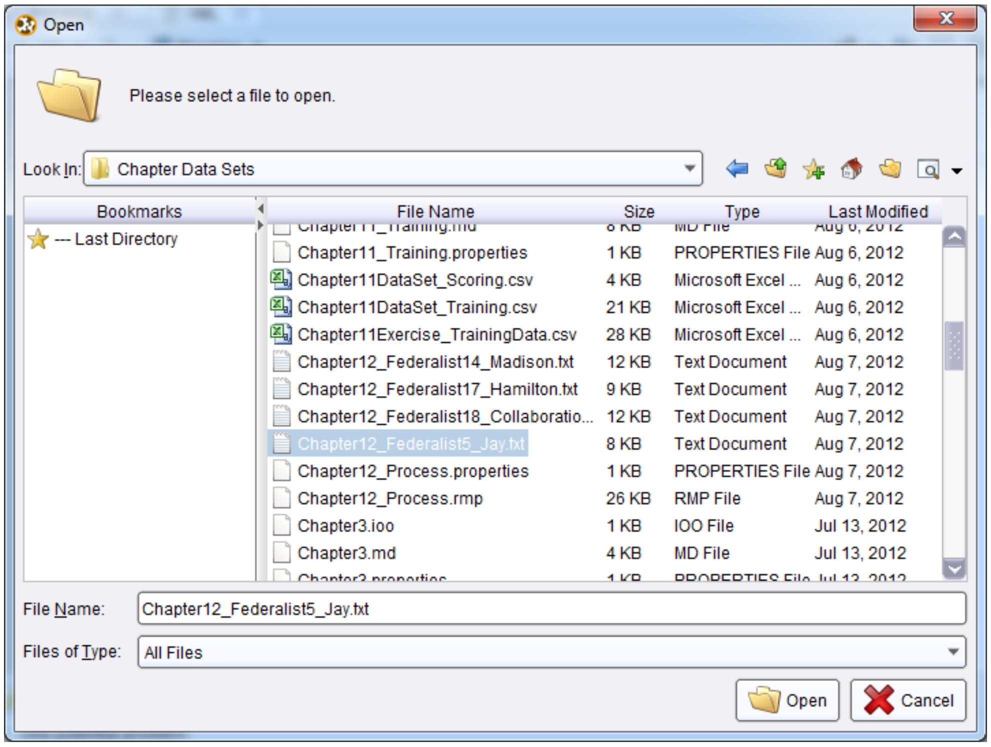


file:
extract text only: ☑
use file extension as type: ☑
encoding: SYSTEM

One potential problem

Figure 12-4. Adding a Read Document operator to our model.

5) In the Parameters area of the RapidMiner window (right hand side), note that you must specify a 'file' that RapidMiner can read. Click on the folder icon to the right of the file parameter to browse for our first text file.



Please select a file to open.

Look In: Chapter Data Sets

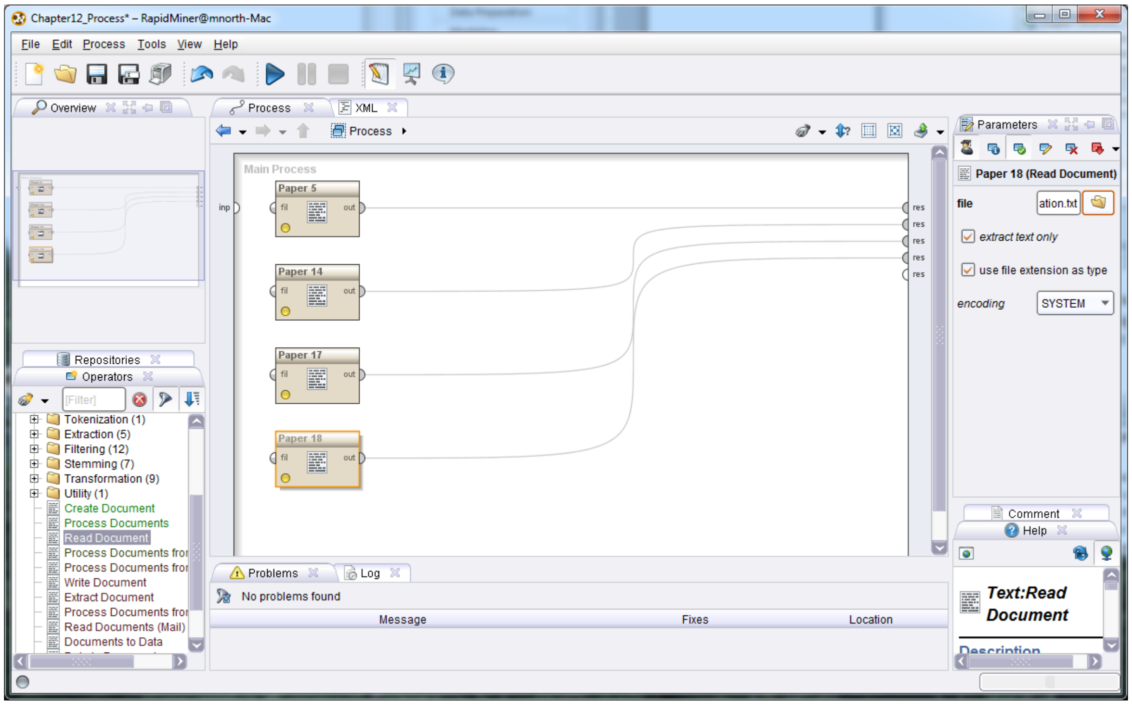
File Name	Size	Type	Last Modified
Chapter11_Training.md	8 KB	MD File	Aug 6, 2012
Chapter11_Training.properties	1 KB	PROPERTIES File	Aug 6, 2012
Chapter11DataSet_Scoring.csv	4 KB	Microsoft Excel ...	Aug 6, 2012
Chapter11DataSet_Training.csv	21 KB	Microsoft Excel ...	Aug 6, 2012
Chapter11Exercise_TrainingData.csv	28 KB	Microsoft Excel ...	Aug 6, 2012
Chapter12_Federalist14_Madison.txt	12 KB	Text Document	Aug 7, 2012
Chapter12_Federalist17_Hamilton.txt	9 KB	Text Document	Aug 7, 2012
Chapter12_Federalist18_Collaboratio...	12 KB	Text Document	Aug 7, 2012
Chapter12_Federalist5_Jay.txt	8 KB	Text Document	Aug 7, 2012
Chapter12_Process.properties	1 KB	PROPERTIES File	Aug 7, 2012
Chapter12_Process.rmp	26 KB	RMP File	Aug 7, 2012
Chapter3.ioo	1 KB	IOO File	Jul 13, 2012
Chapter3.md	4 KB	MD File	Jul 13, 2012
Chapter3.properties	1 KB	PROPERTIES File	Jul 13, 2012

File Name: Chapter12_Federalist5_Jay.txt

Files of Type: All Files

Figure 12-5. Locating the John Jay Federalist Paper (No. 5).

6) In this case, we have saved the text files containing the papers' text in a folder called Chapter Data Sets. We have browsed to this folder, and highlighted the John Jay paper. We can click Open to connect our RapidMiner operator to this text file. This will return us to our main process window in RapidMiner. Repeat steps 4 and 5 three more times, each time connecting one of the other papers, preferably in numerical order, to a Read Document operator in RapidMiner. Use care to ensure that you connect the right operator to the right text file, so that you can keep the text of each paper straight with the operator that's handling it. Once finished, your model should look like Figure 12-6.



Paper 18 (Read Document)

file ation.txt

☑ extract text only

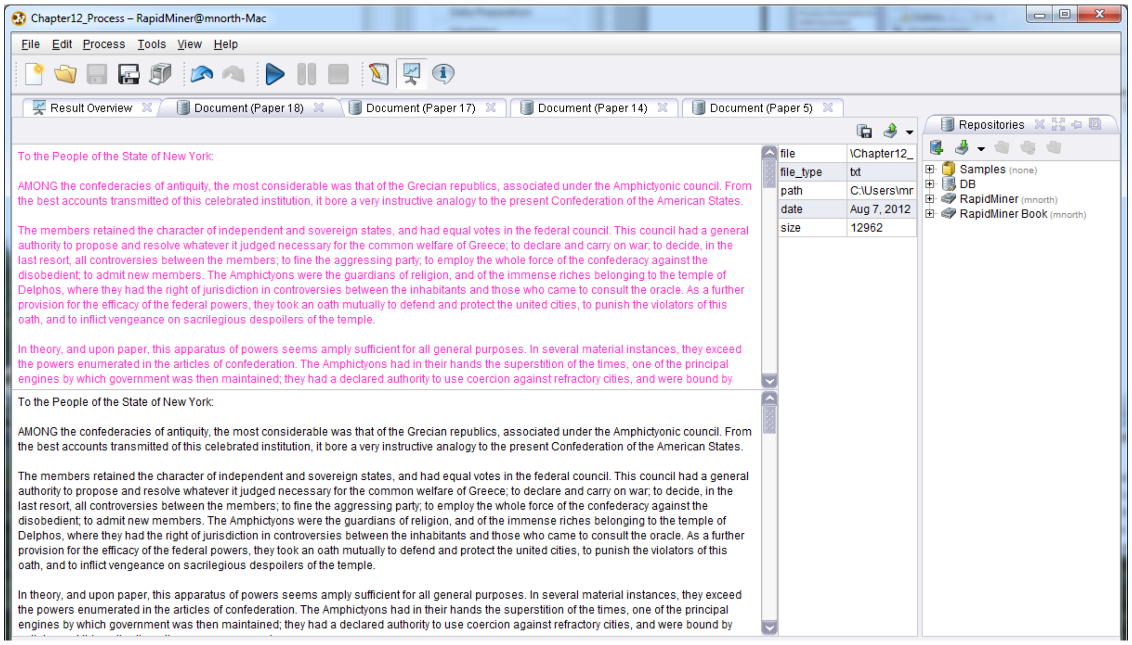
☑ use file extension as type

encoding SYSTEM

No problems found

Figure 12-6. All four Federalist Paper text files are now connected in RapidMiner.

7) Go ahead and run the model. You will see that each of the four papers have been read into RapidMiner and can be reviewed in results perspective. After reviewing the text, return to design perspective.



To the People of the State of New York:

AMONG the confederacies of antiquity, the most considerable was that of the Grecian republics, associated under the Amphictyonic council. From the best accounts transmitted of this celebrated institution, it bore a very instructive analogy to the present Confederation of the American States.

The members retained the character of independent and sovereign states, and had equal votes in the federal council. This council had a general authority to propose and resolve whatever it judged necessary for the common welfare of Greece; to declare and carry on war; to decide, in the last resort, all controversies between the members; to fine the aggressing party; to employ the whole force of the confederacy against the disobedient; to admit new members. The Amphictyons were the guardians of religion, and of the immense riches belonging to the temple of Delphos, where they had the right of jurisdiction in controversies between the inhabitants and those who came to consult the oracle. As a further provision for the efficacy of the federal powers, they took an oath mutually to defend and protect the united cities, to punish the violators of this oath, and to inflict vengeance on sacrilegious despoilers of the temple.

In theory, and upon paper, this apparatus of powers seems amply sufficient for all general purposes. In several material instances, they exceed the powers enumerated in the articles of confederation. The Amphictyons had in their hands the superstition of the times, one of the principal engines by which government was then maintained; they had a declared authority to use coercion against refractory cities, and were bound by

To the People of the State of New York:

AMONG the confederacies of antiquity, the most considerable was that of the Grecian republics, associated under the Amphictyonic council. From the best accounts transmitted of this celebrated institution, it bore a very instructive analogy to the present Confederation of the American States.

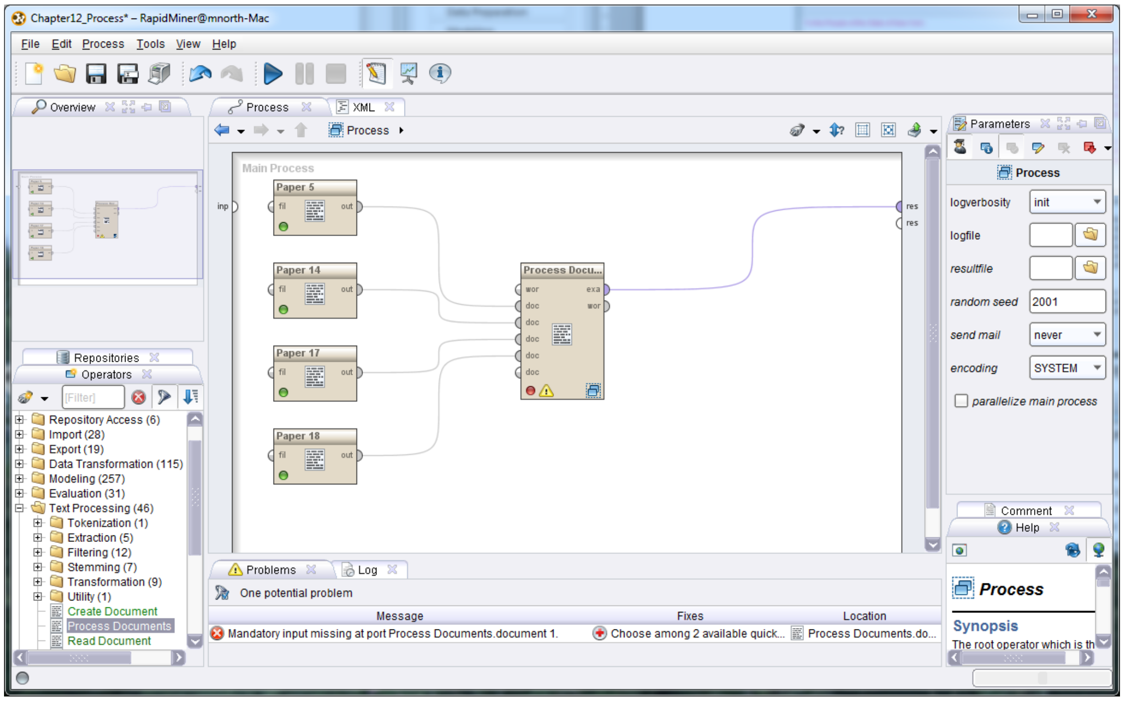
The members retained the character of independent and sovereign states, and had equal votes in the federal council. This council had a general authority to propose and resolve whatever it judged necessary for the common welfare of Greece; to declare and carry on war; to decide, in the last resort, all controversies between the members; to fine the aggressing party; to employ the whole force of the confederacy against the disobedient; to admit new members. The Amphictyons were the guardians of religion, and of the immense riches belonging to the temple of Delphos, where they had the right of jurisdiction in controversies between the inhabitants and those who came to consult the oracle. As a further provision for the efficacy of the federal powers, they took an oath mutually to defend and protect the united cities, to punish the violators of this oath, and to inflict vengeance on sacrilegious despoilers of the temple.

In theory, and upon paper, this apparatus of powers seems amply sufficient for all general purposes. In several material instances, they exceed the powers enumerated in the articles of confederation. The Amphictyons had in their hands the superstition of the times, one of the principal engines by which government was then maintained; they had a declared authority to use coercion against refractory cities, and were bound by

file	\Chapter12_
file_type	txt
path	C:\Users\mr
date	Aug 7, 2012
size	12962

Figure 12-7. Reviewing the suspected collaboration paper (no. 18) in results perspective.

8) We now have our four essays available in RapidMiner. Reading the papers is not enough however. Gillian's goal is to analyze the papers. For this, we will use a Process Documents operator. It is located just above the Read Document operator in the Text Processing menu tree. Drag this operator into your process and drop it into the Paper 5 stream. There will be an empty *doc* port on the bottom left hand side of the Process Documents operator. Disconnect your Paper 14's *out* port from its *res* port and connect it to the open *doc* port instead. Remember that you can rearrange port connections by clicking on the first, then clicking on the second. You will get a warning message asking you to confirm the disconnect/reconnect action each time you do this. Repeat this process until all four documents are feeding into the Process Documents operator, as is the case in Figure 12-8.



Process

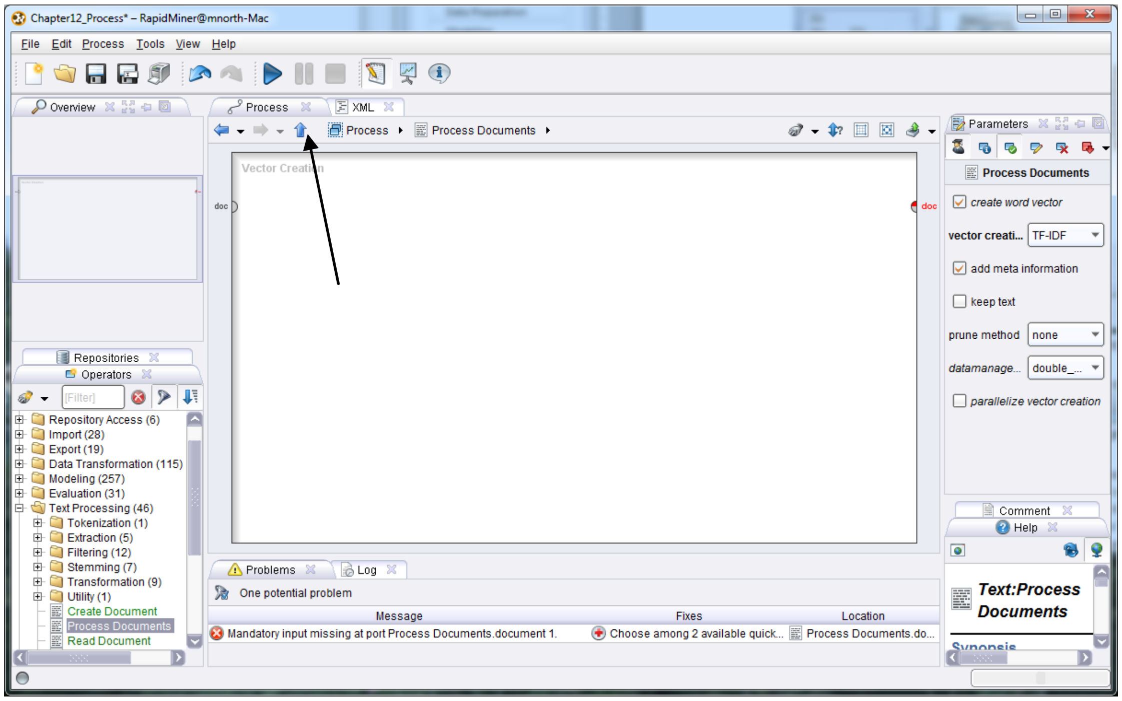
logverbosity	init
logfile	
resultfile	
random seed	2001
send mail	never
encoding	SYSTEM
parallelize main process	☐

One potential problem

Message	Fixes	Location
Mandatory input missing at port Process Documents.document 1.	Choose among 2 available quick...	Process Documents.do...

Figure 12-8. All four Federalist Papers feeding into a single document processor.

9) Next, double click on the Process Documents operator. This will take us into a sub-process window.



Process Documents

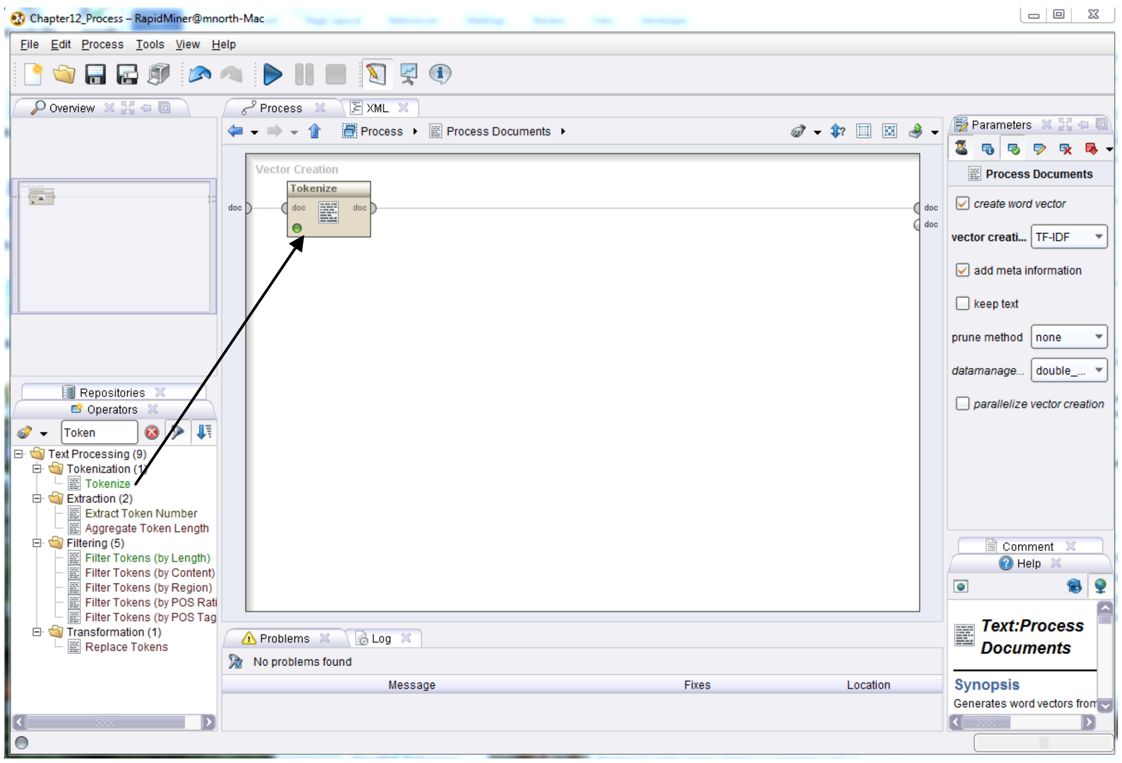
[✓] create word vector
vector creati... TF-IDF
[✓] add meta information
[ ] keep text
prune method none
datamanage... double_...
[ ] parallelize vector creation

One potential problem

Message	Fixes	Location
Mandatory input missing at port Process Documents.document 1.	Choose among 2 available quick...	Process Documents.do...

Figure 12-9. A view inside the sub-process of our Process Documents operator.

10) Note that the blue up arrow in the process toolbar is now illuminated, where previously it has been grayed out. This will allow us to return to our main process, once we have constructed our **sub-process**. Within the sub-process though, there are a few things we *need* to do, and a couple we can choose to do, in order to mine our text. Use the search field in the Operators tab to locate an operator called **Tokenize**. It is under the Text Processing menu in the Tokenization folder. When mining text, the words in the text must be grouped together and counted. Without some numeric structure, the computer cannot assess the meaning of the words. The Tokenize operator performs this function for us. Drag it into the sub-process window (labeled 'Vector Creation' in the upper left hand corner). The *doc* ports from the left hand side of the screen to the operator, and from the operator to the right hand side of the screen, should all be connected by splines, as illustrated in Figure 12-10.



Parameters — Process Documents

- [✓] create word vector
- vector creati... TF-IDF
- [✓] add meta information
- [ ] keep text
- prune method: none
- datamanage... double_...
- [ ] parallelize vector creation

Problems: No problems found

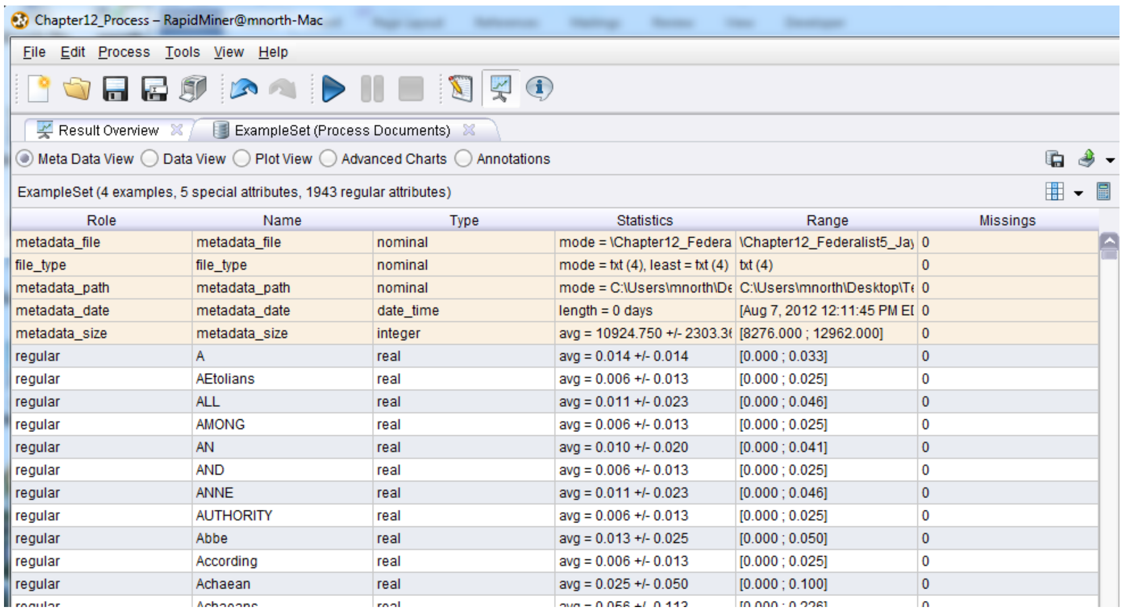
Text:Process Documents

Synopsis

Generates word vectors from

Figure 12-10. Adding tokenization to the text mining model's sub-process.

11) Run the model and briefly review the output. You will see that each word from our four input documents is now an attribute in our data set. We also have a few new special attributes, created by RapidMiner.



ExampleSet (4 examples, 5 special attributes, 1943 regular attributes)

Role	Name	Type	Statistics	Range	Missings
metadata_file	metadata_file	nominal	mode = \Chapter12_Federa	\Chapter12_Federalist5_Ja	0
file_type	file_type	nominal	mode = txt (4), least = txt (4)	txt (4)	0
metadata_path	metadata_path	nominal	mode = C:\Users\mnorth\De	C:\Users\mnorth\Desktop\T	0
metadata_date	metadata_date	date_time	length = 0 days	[Aug 7, 2012 12:11:45 PM E	0
metadata_size	metadata_size	integer	avg = 10924.750 +/- 2303.3	[8276.000 ; 12962.000]	0
regular	A	real	avg = 0.014 +/- 0.014	[0.000 ; 0.033]	0
regular	AEtolians	real	avg = 0.006 +/- 0.013	[0.000 ; 0.025]	0
regular	ALL	real	avg = 0.011 +/- 0.023	[0.000 ; 0.046]	0
regular	AMONG	real	avg = 0.006 +/- 0.013	[0.000 ; 0.025]	0
regular	AN	real	avg = 0.010 +/- 0.020	[0.000 ; 0.041]	0
regular	AND	real	avg = 0.006 +/- 0.013	[0.000 ; 0.025]	0
regular	ANNE	real	avg = 0.011 +/- 0.023	[0.000 ; 0.046]	0
regular	AUTHORITY	real	avg = 0.006 +/- 0.013	[0.000 ; 0.025]	0
regular	Abbe	real	avg = 0.013 +/- 0.025	[0.000 ; 0.050]	0
regular	According	real	avg = 0.006 +/- 0.013	[0.000 ; 0.025]	0
regular	Achaean	real	avg = 0.025 +/- 0.050	[0.000 ; 0.100]	0
regular	Achaeans	real	avg = 0.056 +/- 0.113	[0.000 ; 0.226]	0

Figure 12-11. A view of the words from our input documents as tokens (attributes).

12) Switch back to design perspective. You will see that we return to the sub-process from where we ran the model. We've put the words from our documents into attributes through tokenization, but further processing is needed to make sense of the value of the words in relation to one another. For one thing, there are some words in our data set that really don't mean much. These are necessary conjunctions and articles that make the text readable in English, but that won't tell us much about meaning or authorship. We should remove these words. In the Operators search field, look for the word 'Stop'. These types of words are called **stopwords**, and RapidMiner has built-in dictionaries in several languages to find and filter these out. Add the Filter Stopwords (English) operator to the sub-process stream.

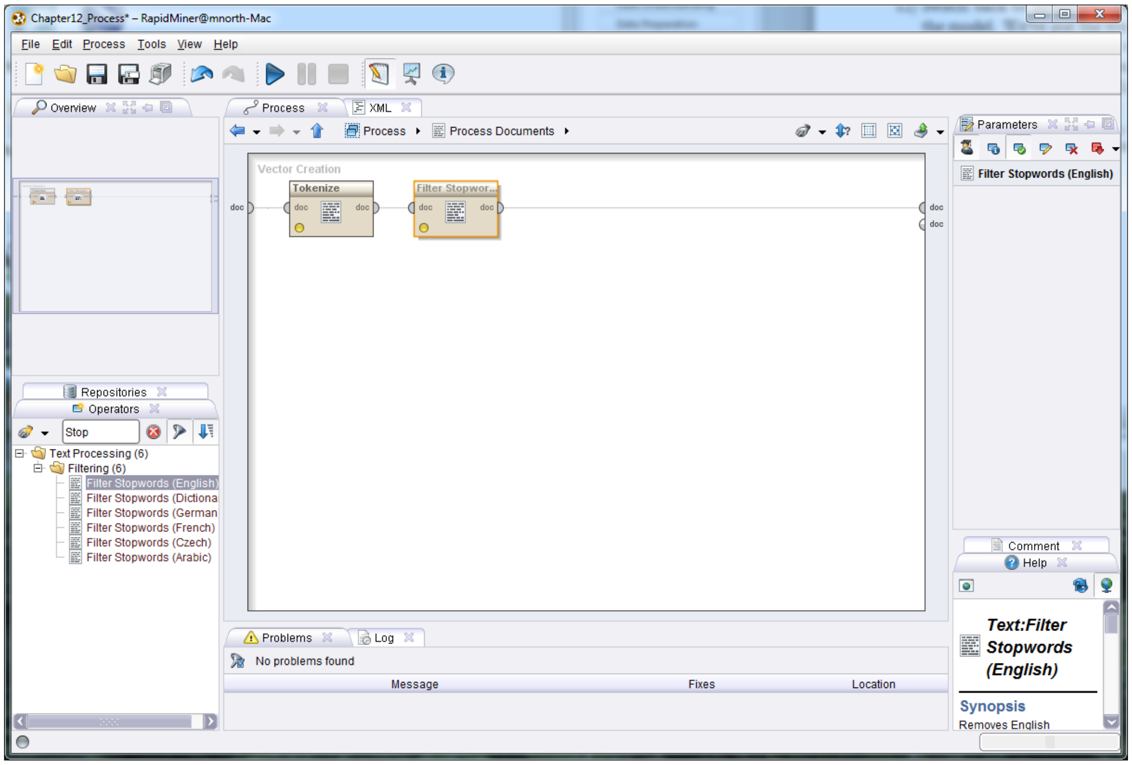
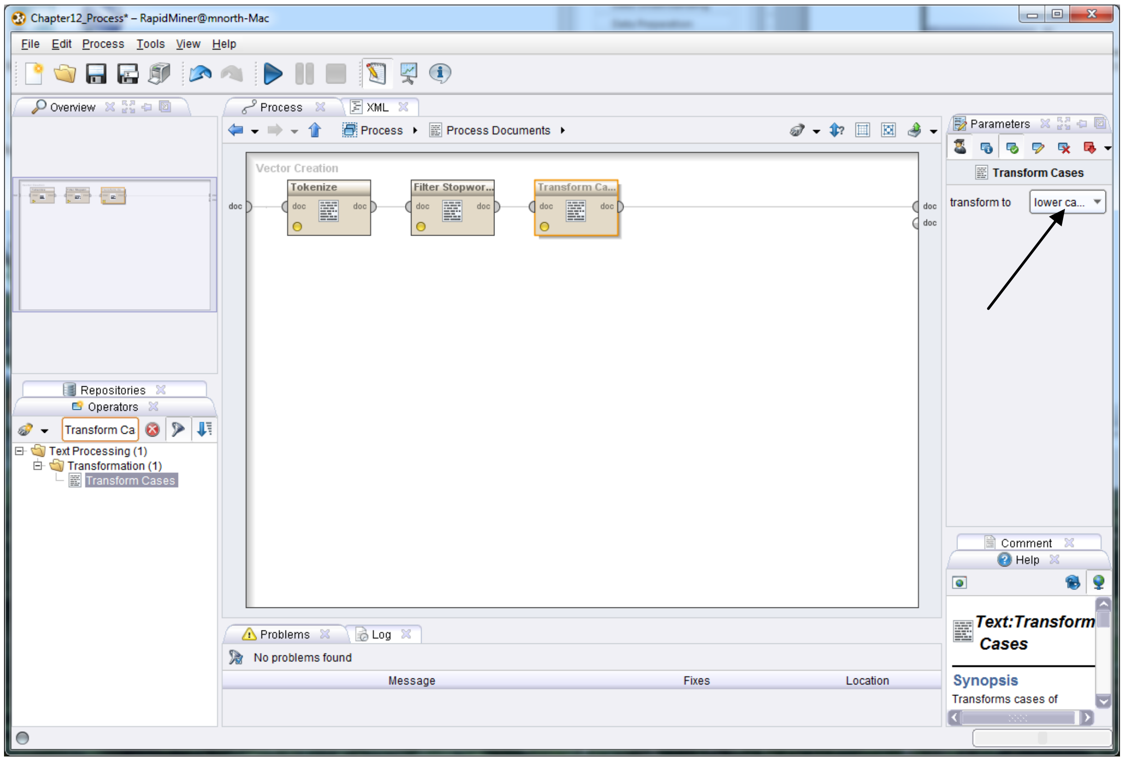


Figure 12-12. Removing stopwords such as 'and', 'or', 'the', etc. from our model.

13) In some instances, letters that are uppercase will not match with the same letters in lowercase. When text mining, this could be a problem because 'Data' might be interpreted different from 'data'. This is known as **Case Sensitivity**. We can address this matter by adding a **Transform Cases** operator to our sub-process stream. Search for this operator in the Operators tab and drag it into your stream, as shown in Figure 12-13.



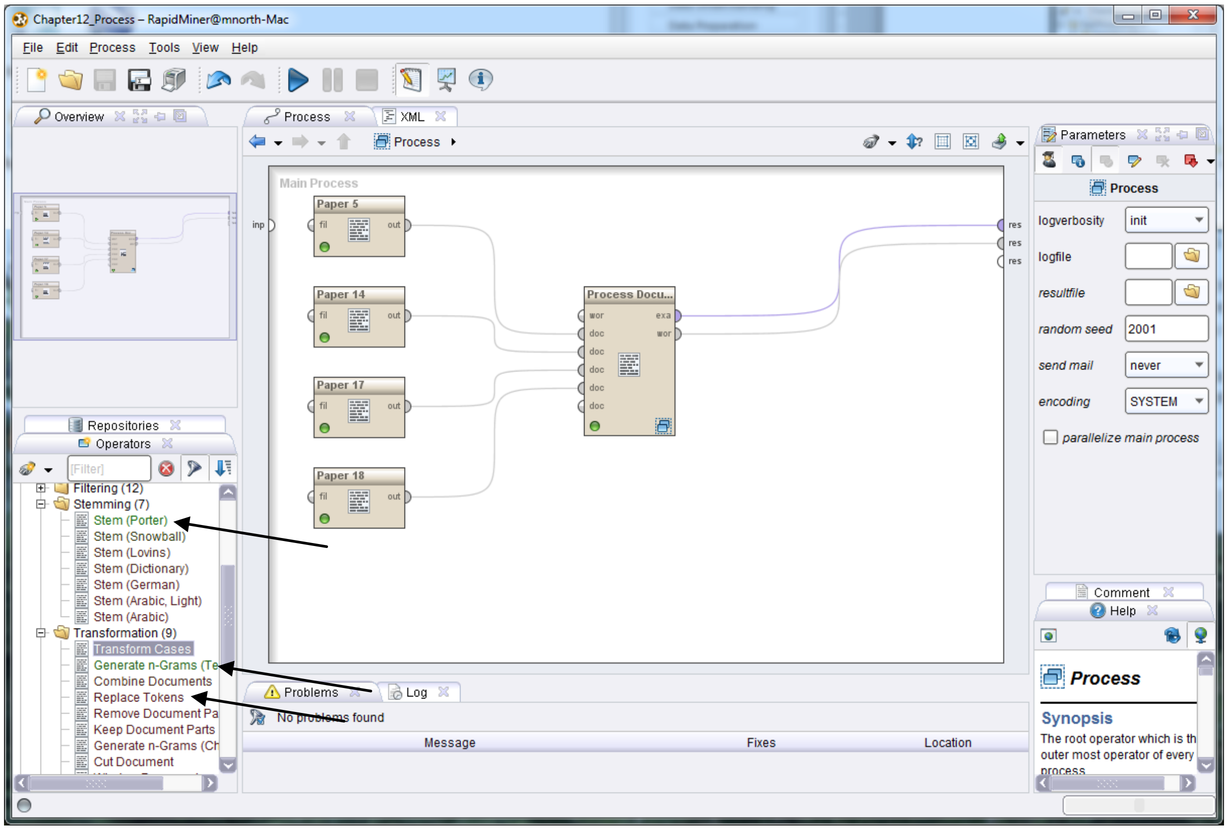
transform to lower ca...

No problems found

Transforms cases of

Figure 12-13. Setting all tokens (word attributes) from our text to be lowercase.

At this point, we have a model that is capable of mining and displaying to us the words that are most frequent in our text documents. This will be interesting for us to review, but there are a few more operators that you should know about in addition to the ones we are using here. These are highlighted by black arrows in Figure 12-14, and discussed below.



Process

logverbosity	init
logfile	
resultfile	
random seed	2001
send mail	never
encoding	SYSTEM

☐ parallelize main process

No problems found

Process

Synopsis

The root operator which is th outer most operator of every process

Figure 12-14. Additional text mining operators of interest.

- **Stemming**: In text mining, stemming means finding terms that share a common root and combining them to mean essentially the same thing. For example, 'America', 'American', 'Americans', are all like terms and effectively refer to the same thing. By stemming (you can see there are a number of stemming operators using different algorithms for you to choose from), RapidMiner can reduce all instances of these word variations to a common form, such as 'Americ', or perhaps 'America', and have all instances represented in a single attribute.

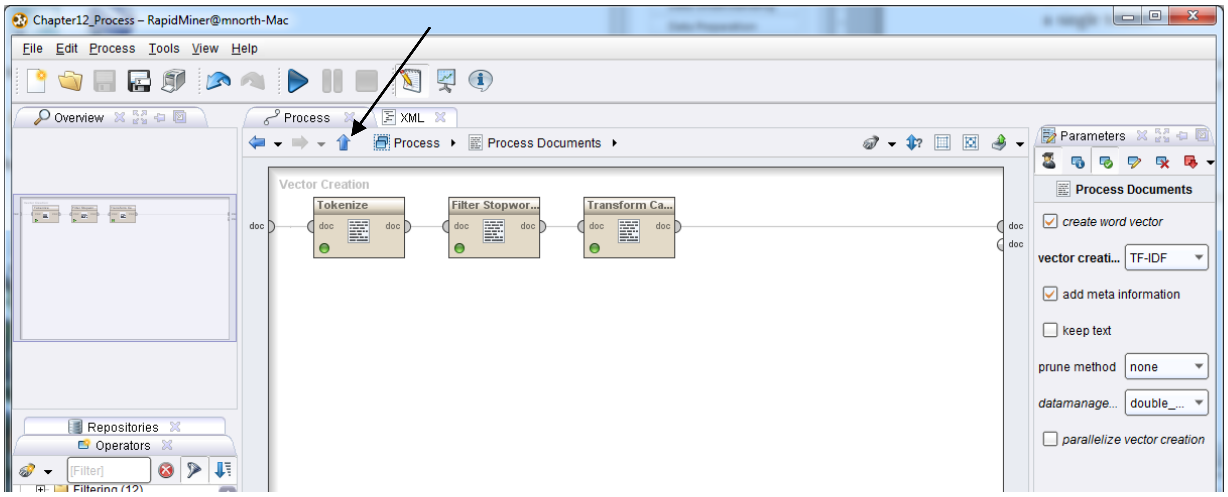
- **Generate n-Grams**: In text mining, an n-gram is a phrase or combination of words that may take on meaning that is different from, or greater than the meaning of each word individually. When creating n-grams, the *n* is simply the maximum number of terms you want RapidMiner to consider grouping together. Take for example the token 'death'. This word by itself is strong, evoking strong emotion. But now consider the meaning, strength and emotion if you were to add a Generate n-Grams operator to your model with a size of 2 (this is set in the parameters area of the n-gram operator). Depending on your input text, you might find the token 'death_penalty'. This certainly has a more specific meaning and evokes different and even stronger emotions than just the token 'death'. What if we increased the n-gram size to 3? We might find a token 'death_penalty_execution'. Again, more specific meaning and perhaps stronger emotion is attached. Understand that these example gram tokens would only be created by RapidMiner if the two or three words in each of them were found together, and in close proximity to one another in the input text. Generating grams can be an excellent way to bring a more granular analysis to your text mining activities.

- **Replace Tokens**: This is similar to replacing missing or inconsistent values in more structured data. This operator can come in handy once you've tokenized your text input. Suppose for example that you had the tokens 'nation', 'country', and 'homeland' in your data set but you wanted to treat all of them as one token. You could use this operator to change both 'country' and 'homeland' to 'nation', and all instances of any of the three terms (or their stems if you also use stemming) would subsequently be combined into a single token.

These are a just a few of the other operators in the Text Processing area that can be nice additions to a text mining model. There are many others, and you may experiment with these at your leisure. For now though, we will proceed to…

MODELING

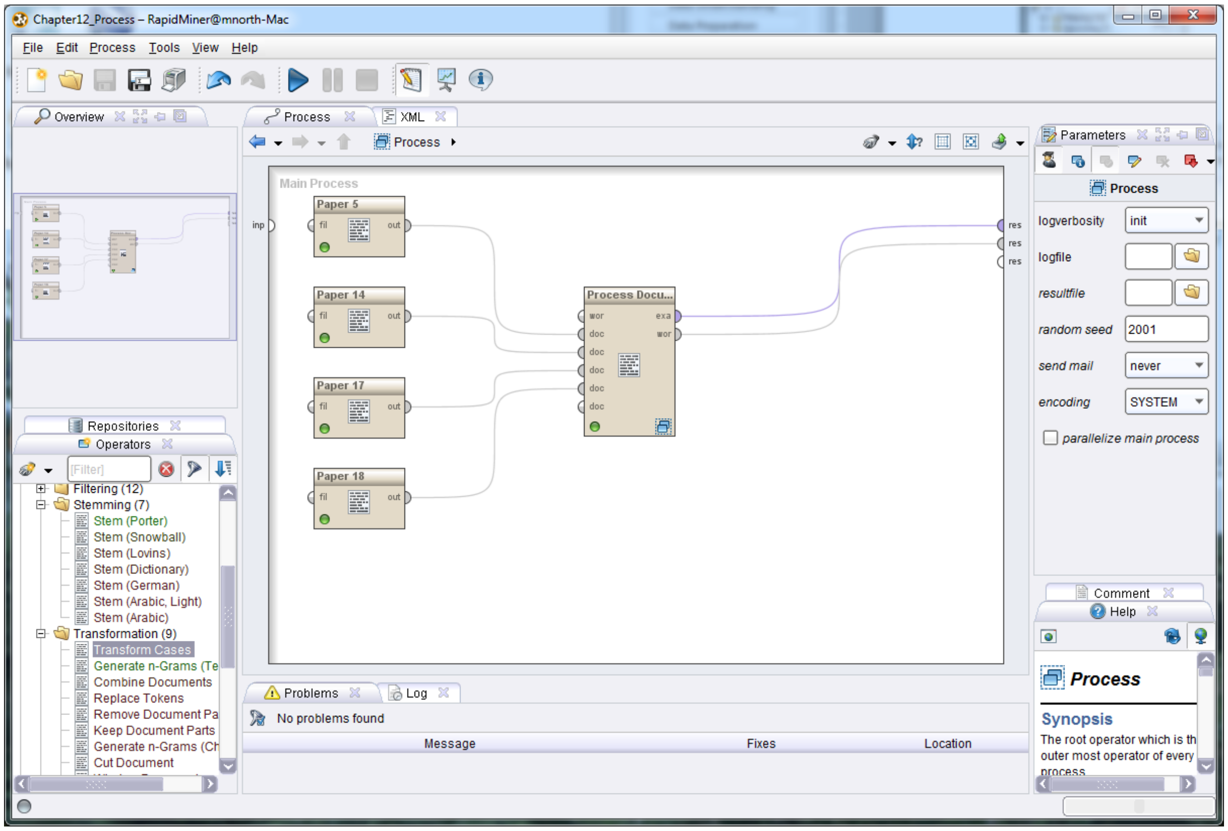
Click the blue up arrow to move from your sub-process back to your main process window.



Process Documents
☑ create word vector
vector creati... TF-IDF
☑ add meta information
☐ keep text
prune method none
datamanage... double_...
☐ parallelize vector creation

Figure 12-15. The 'Return to Parent Operator' arrow (indicated by the black arrow).

In the main process window, ensure that both the *exa* and *wor* ports on the Process Documents operator are connected to *res* ports as shown in Figure 12-16.



logverbosity	init
logfile	
resultfile	
random seed	2001
send mail	never
encoding	SYSTEM
parallelize main process	☐

No problems found

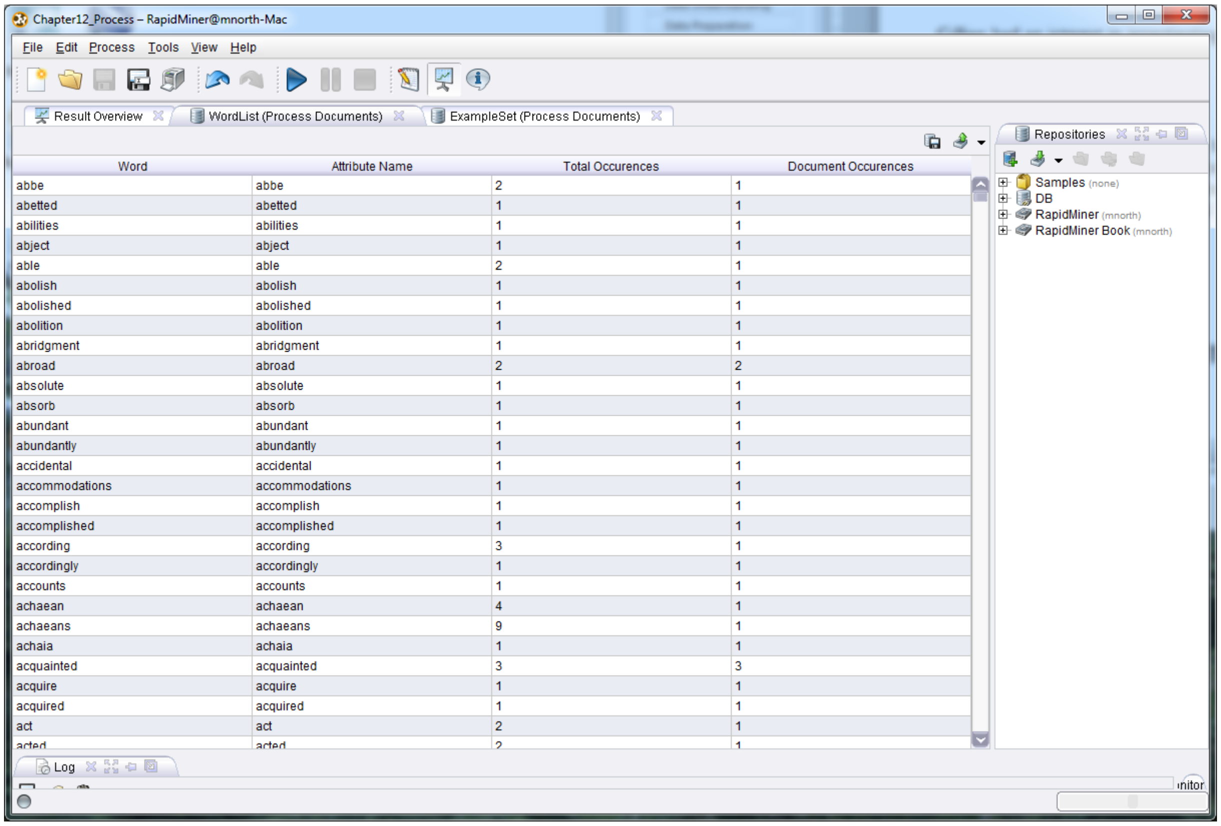
The root operator which is th outer most operator of every process

Figure 12-16. The Federalist Papers text mining model.

The *exa* port will generate a tab in results perspective showing the words (tokens) from our documents as attributes, with the attributes' relative strength in each of the four documents indicated by a decimal coefficient. The *wor* port will create a tab in results perspective that shows the words as tokens with the total number of occurrences, and the number of documents each token appeared in. Although we will do a bit more modeling in this chapter's example, at this point we will go ahead and proceed to…

EVALUATION

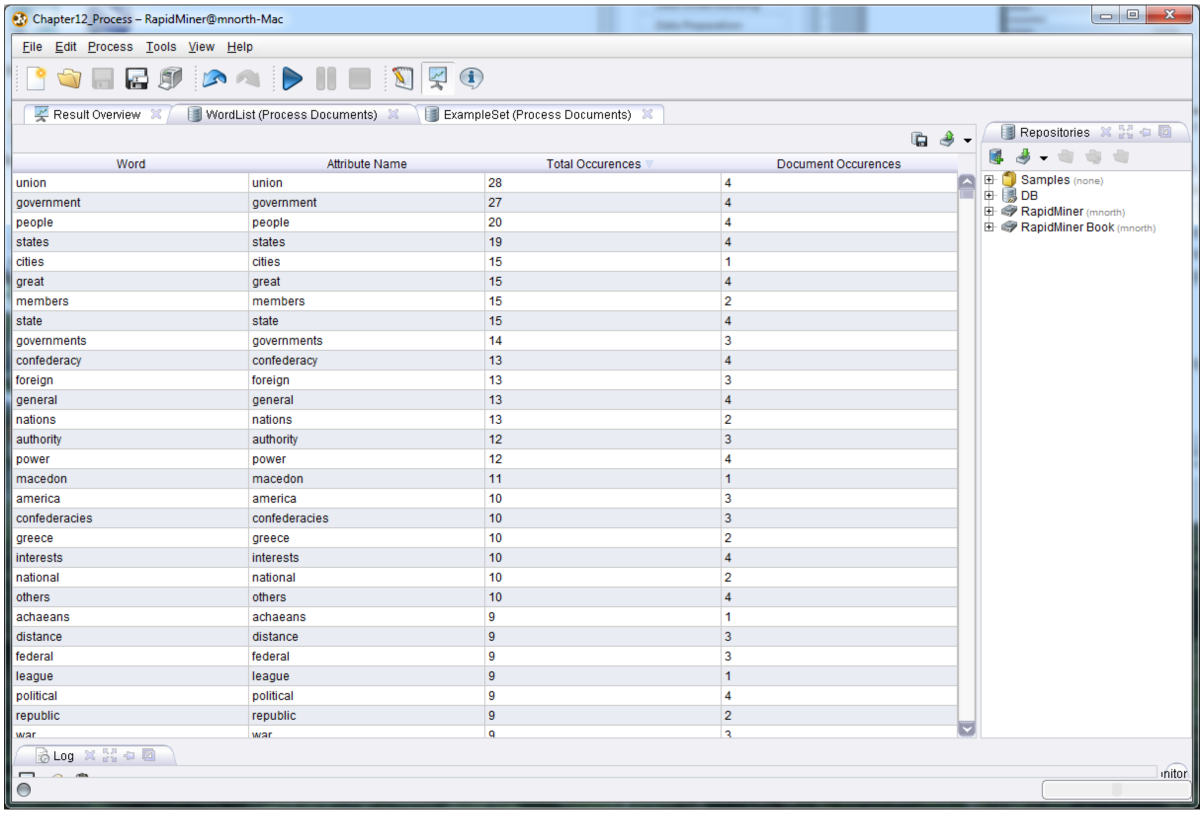
Let's run our model again. We can see the WordList tab in results perspective, showing our tokens and their frequencies in the input documents.


Chapter12_Process – RapidMiner@mnorth-Mac
File Edit Process Tools View Help
Result Overview
WordList (Process Documents)
ExampleSet (Process Documents)
Repositories
Samples (none)
DB
RapidMiner (mnorth)
RapidMiner Book (mnorth)
Log

Word	Attribute Name	Total Occurences	Document Occurences
abbe	abbe	2	1
abetted	abetted	1	1
abilities	abilities	1	1
abject	abject	1	1
able	able	2	1
abolish	abolish	1	1
abolished	abolished	1	1
abolition	abolition	1	1
abridgment	abridgment	1	1
abroad	abroad	2	2
absolute	absolute	1	1
absorb	absorb	1	1
abundant	abundant	1	1
abundantly	abundantly	1	1
accidental	accidental	1	1
accommodations	accommodations	1	1
accomplish	accomplish	1	1
accomplished	accomplished	1	1
according	according	3	1
accordingly	accordingly	1	1
accounts	accounts	1	1
achaean	achaean	4	1
achaeans	achaeans	9	1
achaia	achaia	1	1
acquainted	acquainted	3	3
acquire	acquire	1	1
acquired	acquired	1	1
act	act	2	1
acted	acted	2	1

Figure 12-17. Tokens generated from Federalist Papers 5, 14, 17 and 18, with frequencies.

There are many tokens, which is not surprising considering the length of each essay we have fed into the model. In Figure 12-17, we can see that some of our tokens appear in multiple documents. Consider the word (or token) 'acquainted'. This term shows up one time each in three of the four documents. How can we tell? The Total Occurrences for this token shows as 3, and the Document Occurrences shows as 3, so it must be in each of three documents one time. (Note that even a cursory review of these tokens reveals some stemming opportunities—for example 'accomplish' and 'accomplished' or 'according' and 'accordingly'.) Click on the Total Occurrences column twice to bring the most common terms to the top.


Chapter12_Process – RapidMiner@mnorth-Mac
File Edit Process Tools View Help
Result Overview
WordList (Process Documents)
ExampleSet (Process Documents)
Repositories
Samples (none)
DB
RapidMiner (mnorth)
RapidMiner Book (mnorth)
Log

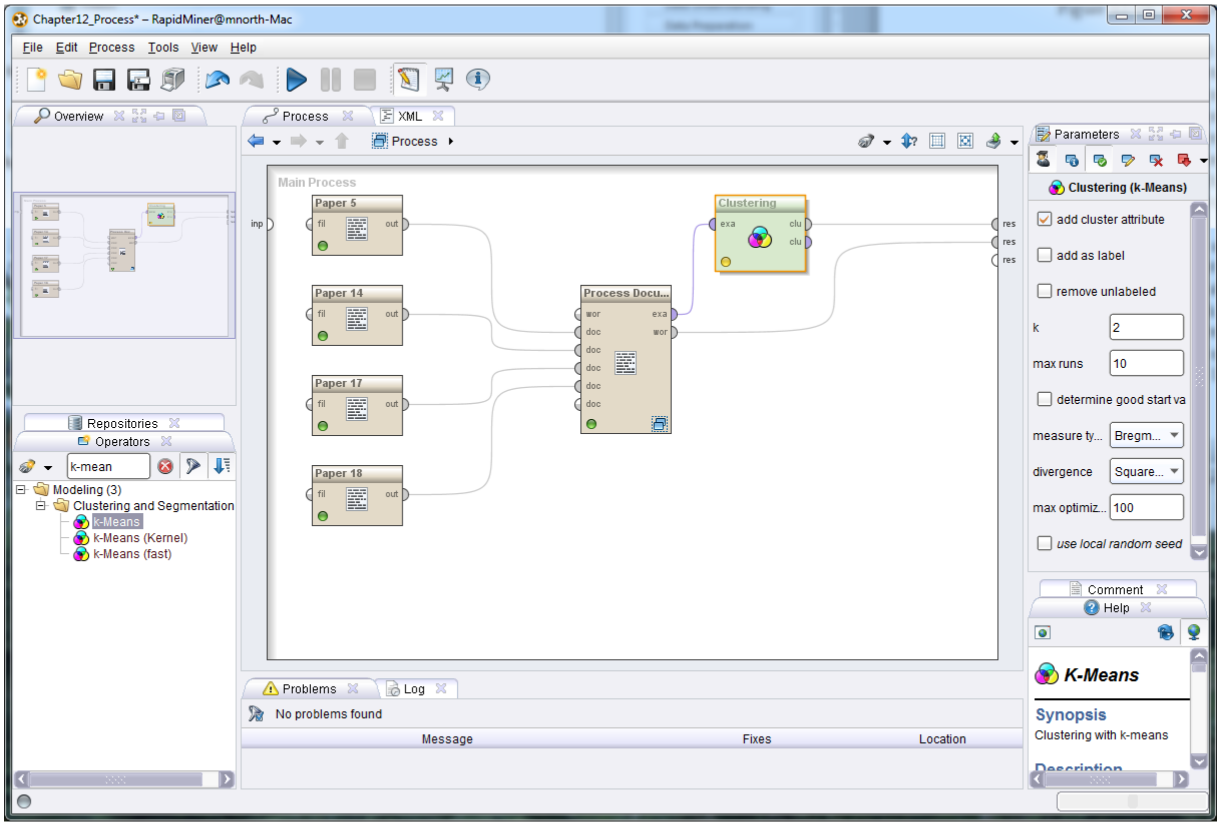
Word	Attribute Name	Total Occurences	Document Occurences
union	union	28	4
government	government	27	4
people	people	20	4
states	states	19	4
cities	cities	15	1
great	great	15	4
members	members	15	2
state	state	15	4
governments	governments	14	3
confederacy	confederacy	13	4
foreign	foreign	13	3
general	general	13	4
nations	nations	13	2
authority	authority	12	3
power	power	12	4
macedon	macedon	11	1
america	america	10	3
confederacies	confederacies	10	3
greece	greece	10	2
interests	interests	10	4
national	national	10	2
others	others	10	4
achaeans	achaeans	9	1
distance	distance	9	3
federal	federal	9	3
league	league	9	1
political	political	9	4
republic	republic	9	2
war	war	9	3

Figure 12-18. Our tokens re-sorted from highest to lowest total occurrences.

Here we see powerful words that all of the authors have relied upon extensively. The Federalist Papers were written to argue in favor of the adoption of a new constitution, and these tokens reflect that agenda. Not only were these terms frequently used across all four documents, the vocabulary reflects the objective of writing and publishing the essays in the first place. Note again here that there is an opportunity to benefit from stemming ('government', 'governments'). Also, some n-grams would be interesting and informative. The term 'great' is both common and frequent, but in what context? Could it be that an n-gram operator might yield the term 'great_nation', which bears much more meaning than just the word 'great'? Feel free to experiment by re-modeling and re-evaluating.

These results in and of themselves are interesting, but we haven't gotten to the heart of Gillian's question, which was: Is it likely that Federalist Paper 18 was indeed a collaboration between Hamilton and Madison? Think back through this book and about what you have learned thus far. We have seen many data mining methodologies that help us to check for affinity or group classifications. Let's attempt to apply one of these to our text mining model to see if it will reveal more about the authors of these papers. Complete the following steps:

1) Switch back to design perspective. Locate the k-Means operator and drop it into your stream between the *exa* port on Process Documents and the *res* port (Figure 12-19).



k-mean

- [✓] add cluster attribute
- [ ] add as label
- [ ] remove unlabeled

k: 2
max runs: 10

- [ ] determine good start va

measure ty...: Bregm...
divergence: Square...
max optimiz...: 100

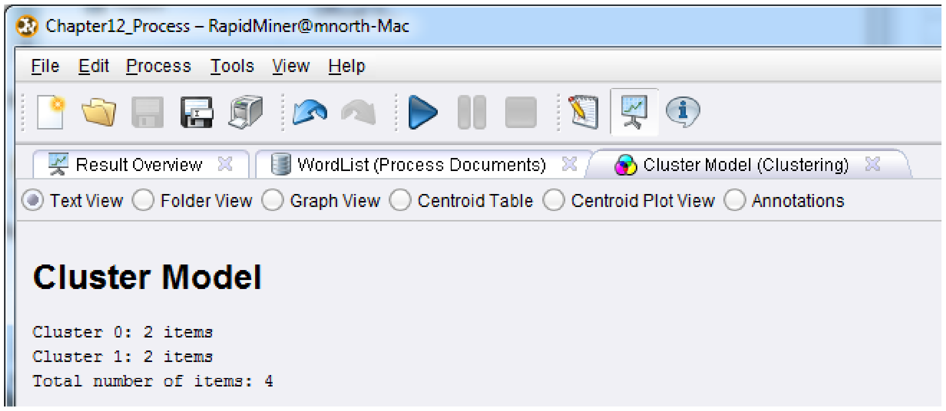
- [ ] use local random seed

No problems found

Message	Fixes	Location

Figure 12-19. Clustering our documents using their token frequncies as means.

2) For this model we will accept the default *k* of 2, since we want to group Hamilton's and Madison's writings together, and keep Jay's separate. We'd hope to get a Hamilton/Madison cluster, with paper 18 in that one, and a Jay cluster with only his paper in there. Run the model and then click on the Cluster Model tab.


Chapter12_Process – RapidMiner@mnorth-Mac
File Edit Process Tools View Help
Result Overview
WordList (Process Documents)
Cluster Model (Clustering)
Text View
Folder View
Graph View
Centroid Table
Centroid Plot View
Annotations

Cluster Model

```
Cluster 0: 2 items
Cluster 1: 2 items
Total number of items: 4
```

Figure 12-20. Cluster results for our four text documents.

3) Unfortunately, it looks like at least one of our four documents ended up associated with John Jay's paper (no. 5). This probably happened for two reasons: (1) We are using the k-Means methodology and means in general tend to try to find a middle with equal parts on both sides; and (2) Jay *was* writing on the same topic as were Hamilton and Madison. Thus, there is going to be much similarity across the essays, so the means will more easily balance even if Jay didn't contribute to paper 18. The topic alone will cause enough similarity that paper 18 could be grouped with Jay, especially when the operator we've chosen is trying to find equal balance. We can see how the four papers have been clustered by clicking on the Folder View radio button and expanding both of the folder menu trees.

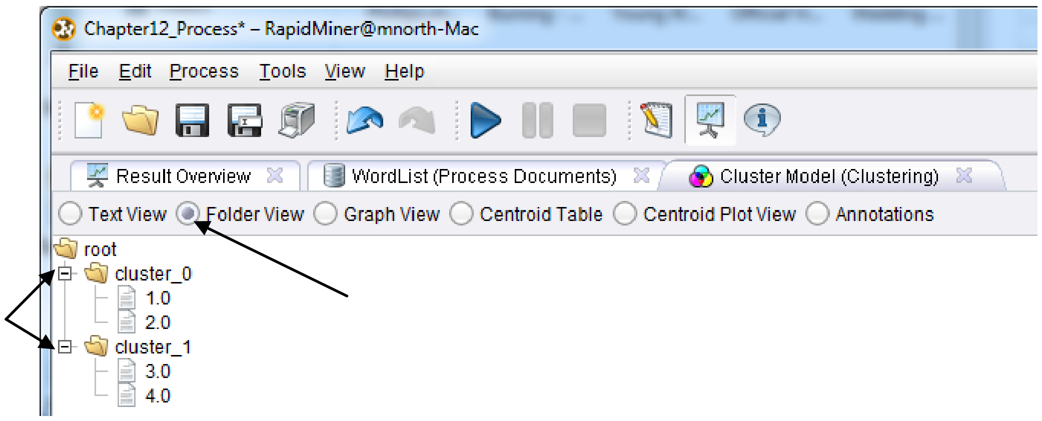
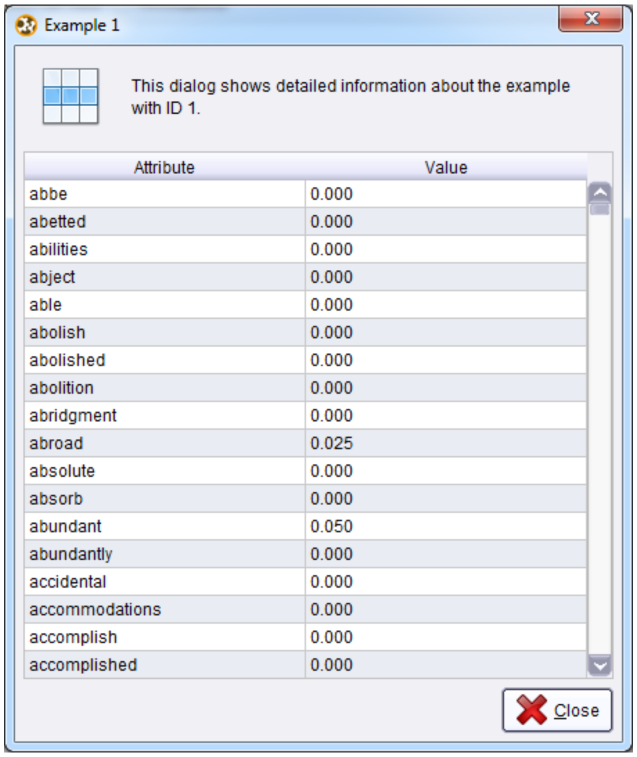

Chapter12_Process* – RapidMiner@mnorth-Mac
File Edit Process Tools View Help
Result Overview
WordList (Process Documents)
Cluster Model (Clustering)
Text View
Folder View
Graph View
Centroid Table
Centroid Plot View
Annotations
root
cluster_0
1.0
2.0
cluster_1
3.0
4.0

Figure 12-21. Examining the document clusters.

4) We can see that the first two papers and the last two papers were grouped together. This can be a bit confusing because RapidMiner has renumbered the documents from 1 to 4, in the order that we added them to our model. In the book's example, we added them in numerical order: 5, 14, 17, and then 18. So paper 5 corresponds to document 1, paper 14 corresponds to document 2, and so forth. If we can't remember the order in which we added the papers to the model, we can click on the little white page icon to the left of the document number to view the document's details:

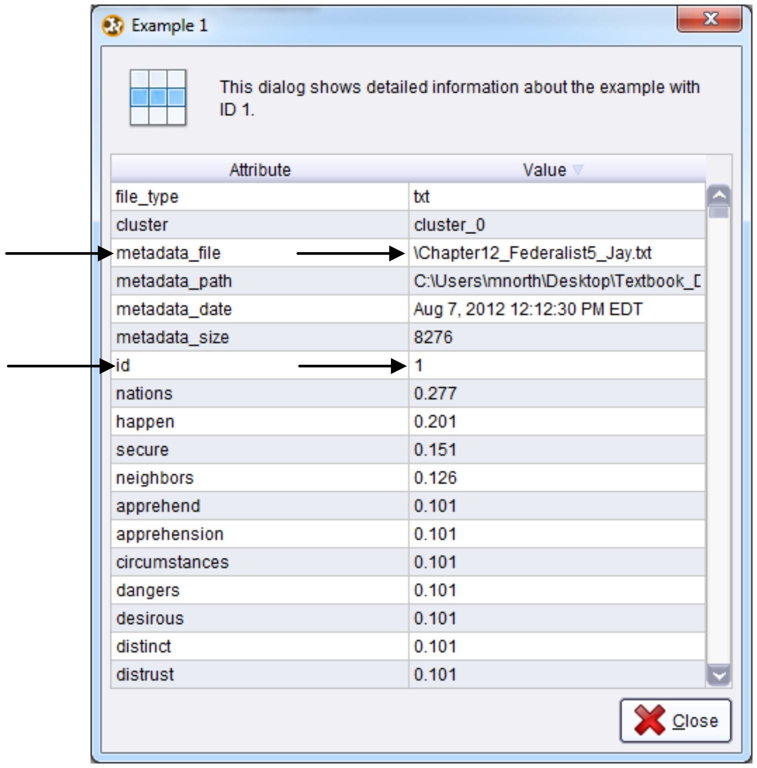


This dialog shows detailed information about the example with ID 1.

Attribute	Value
abbe	0.000
abetted	0.000
abilities	0.000
abject	0.000
able	0.000
abolish	0.000
abolished	0.000
abolition	0.000
abridgment	0.000
abroad	0.025
absolute	0.000
absorb	0.000
abundant	0.050
abundantly	0.000
accidental	0.000
accommodations	0.000
accomplish	0.000
accomplished	0.000

Figure 12-22. Details of document 1.0 in RapidMiner.

5) Click on the Value column heading twice. This will bring the file path for the document toward the top, as shown in Figure 12-23.



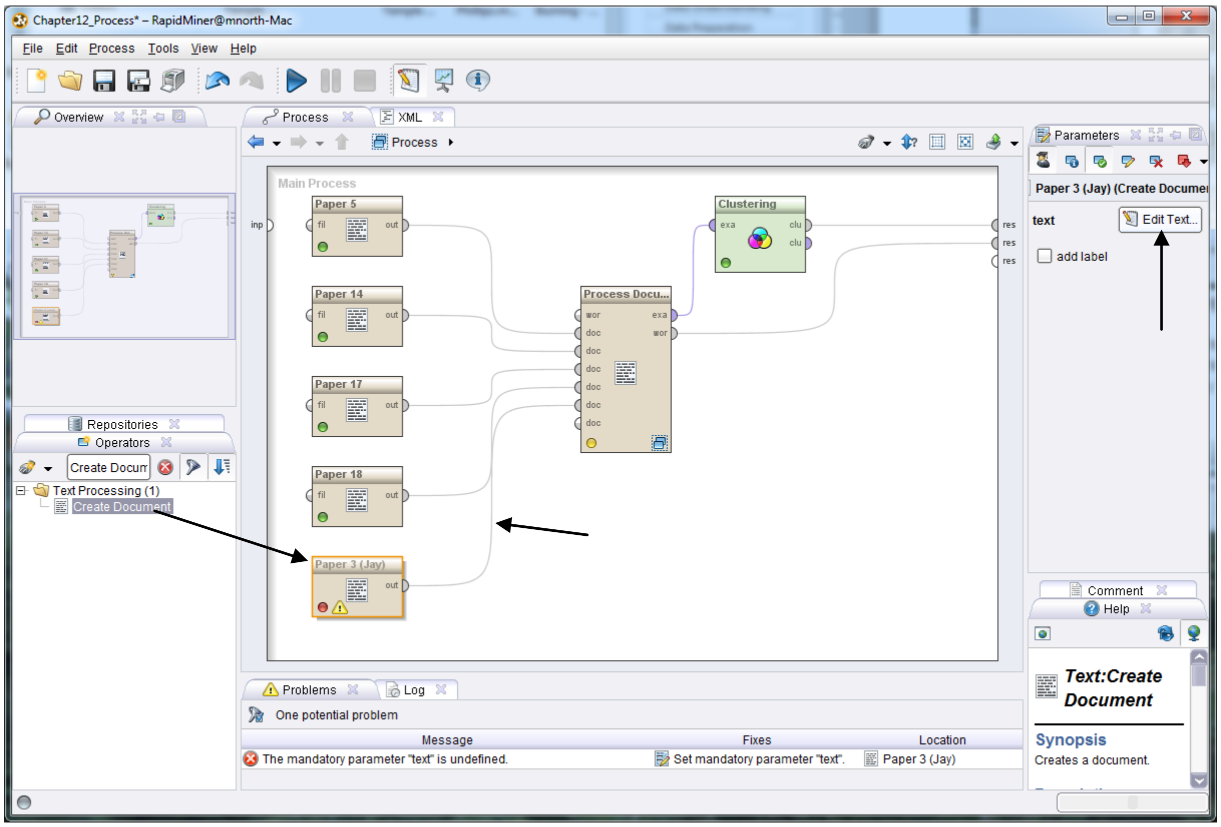
This dialog shows detailed information about the example with ID 1.

Attribute	Value
file_type	txt
cluster	cluster_0
metadata_file	\Chapter12_Federalist5_Jay.txt
metadata_path	C:\Users\mnorth\Desktop\Textbook_[
metadata_date	Aug 7, 2012 12:12:30 PM EDT
metadata_size	8276
id	1
nations	0.277
happen	0.201
secure	0.151
neighbors	0.126
apprehend	0.101
apprehension	0.101
circumstances	0.101
dangers	0.101
desirous	0.101
distinct	0.101
distrust	0.101

Figure 12-23. Document 1's values in reverse sort order.

6) We can see by looking at the first several attributes that for document ID 1, the file is Chapter12_Federalist05_Jay.txt. Thus if we can't remember that we added paper 5 first, resulting in RapidMiner labeling it document 1, we can check it in the document details. This little trick works when you have used the Read Document operator, as the document being read becomes the value for the metadata_file attribute, however when using some other operators, such as the Create Document operator, it doesn't work, as you will see momentarily. Since we added our papers in numerical order in this chapter's example, we do not necessarily need to view and sort the details for each of the documents, but you may if you wish. Knowing that documents 1 and 2 are Jay (no. 5) and Madison (no. 14), and documents 3 and 4 are Hamilton (no. 17) and suspected collaboration (no. 18), we can be encouraged by what we see in this model. It appears that Hamilton *does* have something to do with Federalist Paper 18, but we don't know about Madison yet because Madison was grouped with Jay, probably as a result of the previously discussed mean balancing that k-means clustering is prone to do.

7) Perhaps we can address this by better training our model to recognize Jay's writing. Using your favorite search engine, search the Internet for the text of Federalist Paper No. 3. Gillian knows that this paper's authorship has been connected to John Jay. We will use the text to train our model to better recognize Jay's writing. If paper 18 was written by, or even contributed to by Jay, perhaps we will find that it gets clustered with Jay's papers 3 and 5 when we add paper 3 to the model. In this case, Hamilton and Madison should get clustered together. If on the other hand paper 18 was *not* written or contributed to by Jay, paper 18 should gravitate toward Hamilton (no. 17) and/or Madison (no. 14), so long as Jay was consistent in his writing between papers 3 and 5. Copy the text of paper 3 by highlighting it in whichever web site you found (it is available on a number of sites). Then in design perspective in RapidMiner, locate the Create Document operator and drag it into your process (Figure 12-23).



One potential problem

Message	Fixes	Location
The mandatory parameter "text" is undefined.	Set mandatory parameter "text".	Paper 3 (Jay)

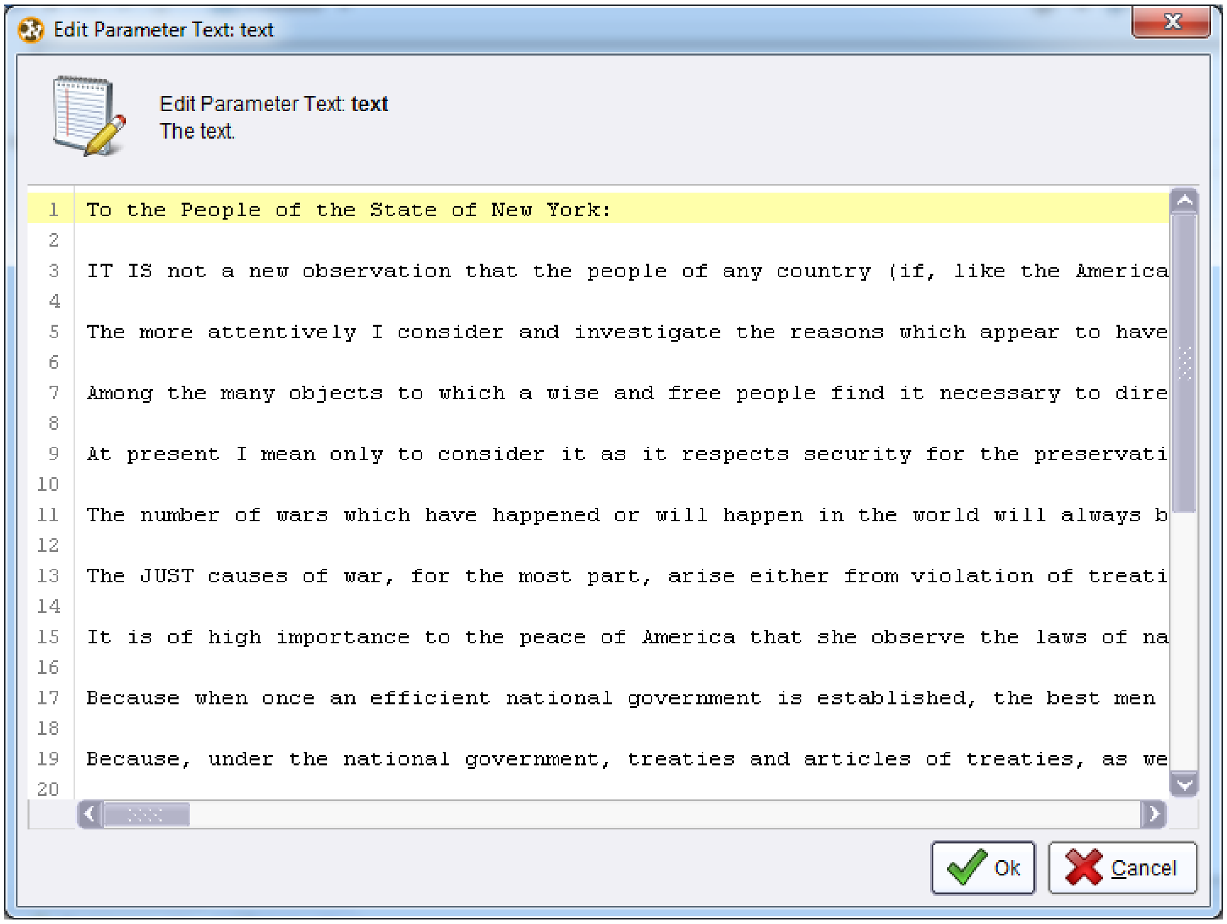
Text:Create Document

Synopsis

Creates a document.

Figure 12-23. Adding a Create Document operator to our text mining model.

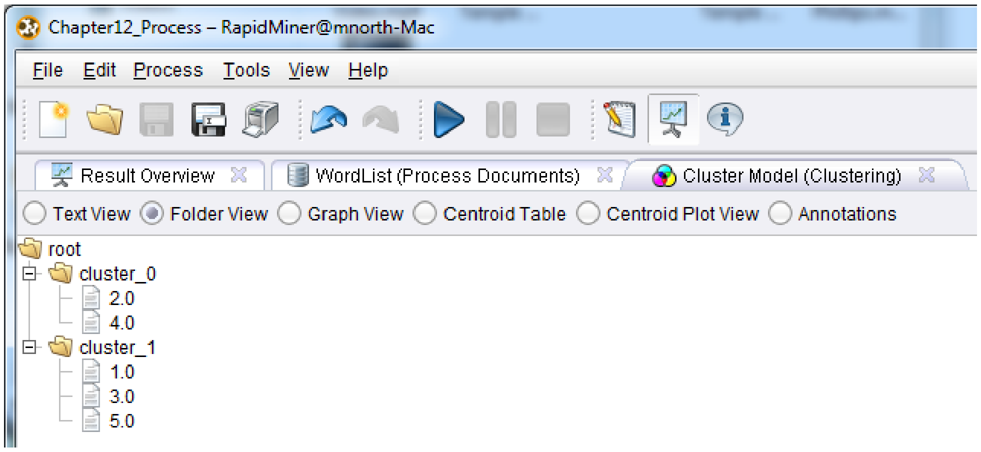
8) Be sure the Create Document operator's *out* port is connected to one of the Process Document operator's *doc* ports. It will likely connect itself to a *res* port, so you'll have to reconnect it to the Process Documents operator. Let's rename this operator 'Paper 3 (Jay)'. Then click on the Edit Text button in the Parameters area on the right hand side of the screen. You will see a window like Figure 12-24.



```
To the People of the State of New York:

IT IS not a new observation that the people of any country (if, like the America

The more attentively I consider and investigate the reasons which appear to have

Among the many objects to which a wise and free people find it necessary to dire

At present I mean only to consider it as it respects security for the preservati

The number of wars which have happened or will happen in the world will always b

The JUST causes of war, for the most part, arise either from violation of treati

It is of high importance to the peace of America that she observe the laws of na

Because when once an efficient national government is established, the best men

Because, under the national government, treaties and articles of treaties, as we

```

Figure 12-24. Adding a text document through a Create Document operator.

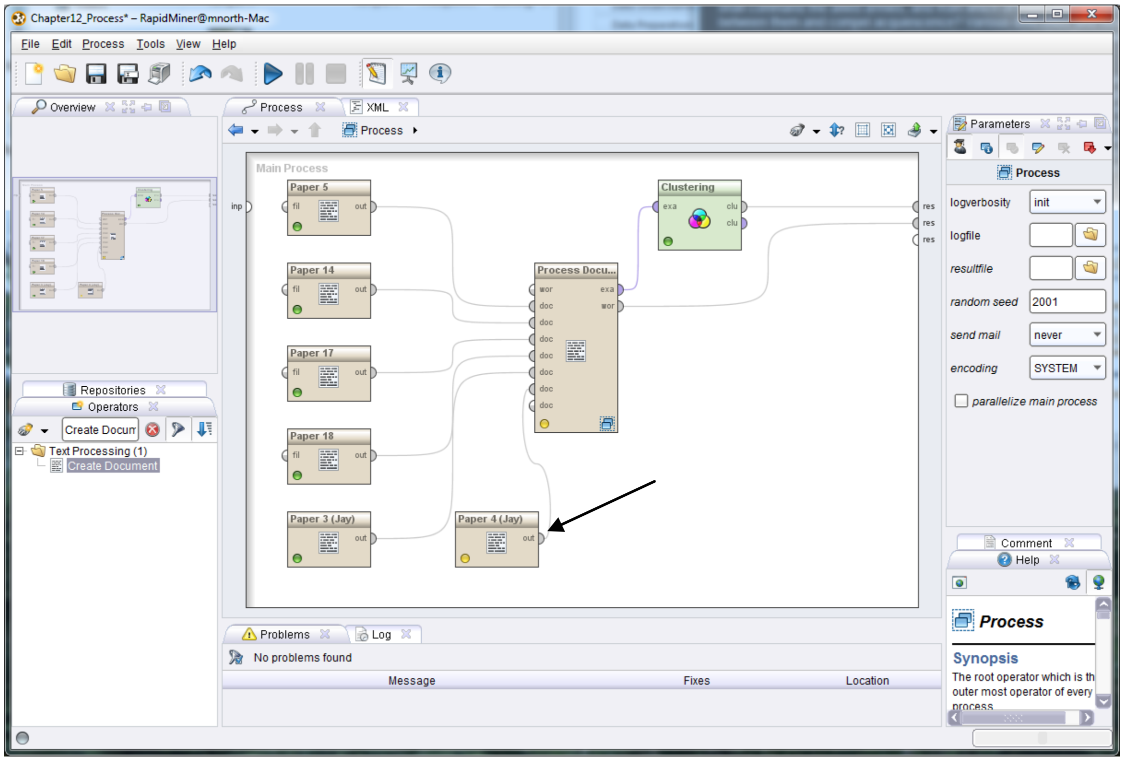
9) Paste the text of Federalist Paper 3 into the Edit Parameter Text window and then click OK. We now have five documents to be processed and run through our k-Means model. RapidMiner will assign document ID 5 to this new document, since it was the fifth one we added to our main process. Let's run the model to see how our documents are grouped now.



```
root
  cluster_0
    2.0
    4.0
  cluster_1
    1.0
    3.0
    5.0
```

Figure 12-25. New clusters identified by RapidMiner with the addition of another of Jay's papers.

10) On the Cluster Model tab in results perspective, with the cluster menu trees expanded, we now see that documents 2 and 4 (papers 14 (Madison) and 18 (collaboration)) are grouped together, while the two of Jay's papers (documents 1 (paper 5) and 5 (paper 3)) are grouped with Hamilton's paper (document 3; paper 17). This is very encouraging because the suspected collaboration paper (no. 18) has now been associated with both Madison's and Hamilton's writing, but not with Jay's. Let's give our model one more of Jay's papers to further train it in Jay's writing style, and see if we can find further evidence that paper 18 is most strongly connected to Madison and Hamilton. Repeat steps 7 through 9, only this time, find the text of Federalist Paper 4 (also written by John Jay) and paste it into a new Create Document operator.



logverbosity: init
logfile
resultfile
random seed: 2001
send mail: never
encoding: SYSTEM
parallelize main process

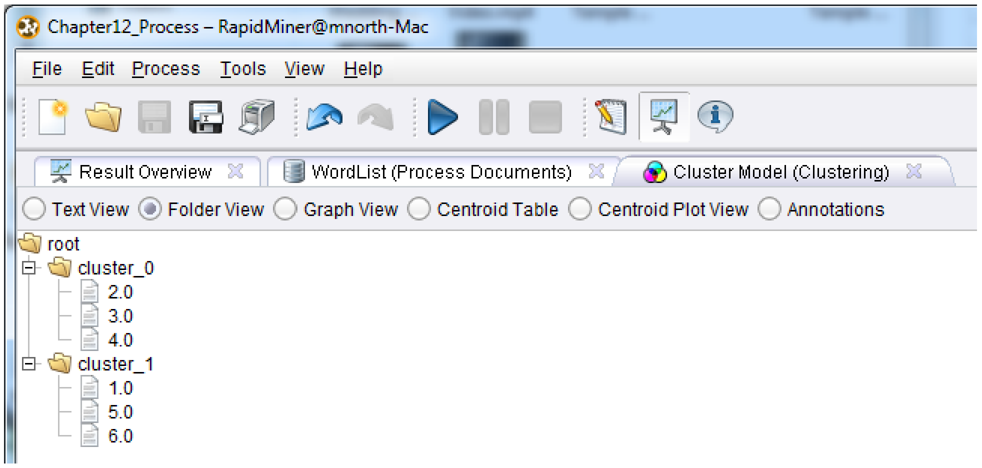
No problems found

Figure 12-26. The addition of another Create Document operator containing the text of Federalist Paper 4 by John Jay.

11) Be sure to rename the second Create Document operator descriptively, as we have done in Figure 12-26. When you have used the Edit Text button to paste the text for Federalist Paper 4 into your model and have ensured that your ports are all connected correctly, run the model one last time and we will proceed to…

DEPLOYMENT

Gillian had an interest in investigating the similarities and differences between several of the Federalist Papers in order to lend credence to the belief that Alexander Hamilton and James Madison collaborated on paper 18.


Chapter12_Process – RapidMiner@mnorth-Mac
File Edit Process Tools View Help
Result Overview
WordList (Process Documents)
Cluster Model (Clustering)
Text View
Folder View
Graph View
Centroid Table
Centroid Plot View
Annotations

```
root
  cluster_0
    2.0
    3.0
    4.0
  cluster_1
    1.0
    5.0
    6.0
```

Figure 12-27. Final cluster results after training our text mining model to recognize John Jay's writing style.

Gillian now has the evidence she had hoped to find. As we continued to train our model in John Jay's writing style, we have found that he indeed was consistent from paper 3 to 4 to 5, as RapidMiner found these documents to be the most similar and subsequently clustered them together in cluster_1. At the same time, RapidMiner consistently found paper 18, the suspected collaboration between Hamilton and Madison to be associated with one, then the other, and finally both of them together. Gillian could further strengthen her model by adding additional papers from all three authors, or she could go ahead and add what we've already found to her exhibit at the museum.

CHAPTER SUMMARY

Text mining is a powerful way of analyzing data in an unstructured format such as in paragraphs of text. Text can be fed into a model in different ways, and then that text can be broken down into tokens. Once tokenized, words can be further manipulated to address matters such as case sensitivity, phrases or word groupings, and word stems. The results of these analyses can reveal the frequency and commonality of strong words or grams across groups of documents. This can reveal trends in the text, such as what topics are most important to author(s), or what message should be taken away from the text when reading the documents.

Further, once the documents' tokens are organized into attributes, the documents can be modeled, just as other, more structured data sets can be modeled. Multiple documents can be handled by a single Process Document operator in RapidMiner, which will apply the same set of tokenization and token handlers to all documents at once through the sub-process stream. After a model has been applied to a set of documents, additional documents can be added to the stream, passed through the document processor, and run through the model to yield more well-trained and specific results.

REVIEW QUESTIONS

1) What are some of the benefits of text mining as opposed to the other models you've learned in this book?

2) How are some ways that text-based data is imported into RapidMiner?

3) What is a sub-process and when do you use one in RapidMiner?

4) Define the following terms: token, stem, n-gram, case-sensitive.

5) How does tokenization enable the application of data mining models to text-based data?

6) How do you view a k-Means cluster's details?

EXERCISE

For this chapter's exercise, you will mine text for common complaints against a company or industry. Complete the following steps.

1) Using your favorite search engine, locate a web site or discussion forum on the Internet where people have posted complaints, criticisms or pleas for help regarding a company or an industry (e.g. airlines, utility companies, insurance companies, etc.).
2) Copy and paste at least ten of these posts or comments into a text editor, saving each one as its own text document with a unique name.
3) Open a new, blank process in RapidMiner, and using the Read Documents operator, connect to each of your ten (or more) text documents containing the customer complaints you found.
4) Process these documents in RapidMiner. Be sure you tokenize and use other handlers in your sub-process as you deem appropriate/necessary. Experiment with grams and stems.
5) Use a k-Means cluster to group your documents into two, three or more clusters. Output your word list as well.
6) Report the following:
    a. Based on your word list, what seem to be the most common complaints or issues in your documents? Why do you think that is? What evidence can you give to support your claim?
    b. Based on your word list, are there some terms or phrases that show up in all, or at least most of your documents? Why do you think these are so common?
    c. Based on your clusters, what groups did you get? What are the common themes in each of your clusters? Is this surprising? Why or why not?
    d. How might a customer service manager use your model to address the common concerns or issues you found?

Challenge Step!

7) Using your knowledge from past chapters, removed the k-Means clustering operator, and try to apply a different data mining methodology such as association rules or decision trees to your text documents. Report your results.