Chapter 9: Introduction to Predictive Modeling with Textual Data

- Chapter 9 gives an introduction to predictive modeling using unstructured textual data.
- Quantifying textual data and putting it into a spread sheet or SAS table form is an important pre-requisite for developing predictive models with textual data.

Quantifying textual data involves several steps:

These are parsing the documents, filtering, and reducing decreasing the dimension reduction. Dimension reduction is done by Singular Value Decomposition (SVD).

- We first illustrated the quantification of textual data, Boolean Retrieval method and dimension reduction using SVD method using a simplified example.
- Then we showed how to use Text Parsing, Text Filter, Text Topic and Text Cluster nodes.
- Then we illustrated how to use the output data set produced by the Text Topic node for estimating a logistic regression equation.
- Using a simple example I demonstrated the Expectation-Maximization (EM) Clustering.
- We have explained the Hierarchical clustering method with simple algebra.

Introduction

- This chapter shows how you can use SAS Enterprise Miner's text mining nodes1 to quantify unstructured textual data and create data matrices and SAS data sets that can be used for statistical analysis and predictive modeling. Some examples of textual data are: web pages, news articles, research papers, insurance reports, etc.
- In text mining, each web page, news article, research paper, etc. is called a document.
- A collection of documents, called a corpus of documents, is required for developing predictive models or classifiers. Each document is originally represented by a string of characters.
- In order to build predictive models from this data, you need to represent each document by a numeric vector whose elements are frequencies or some other measure of occurrence of different terms in the document.
- Quantifying textual information is nothing but converting the string of characters in a document to a numerical vector of frequencies of occurrences of different words or terms.
- The numerical vector is usually a column in the term-document matrix, whose columns represent the documents. A data matrix is the transpose of the term-document matrix.
- You can attach a target column with known class labels of the documents to the data matrix. Examples of class labels are: Automobile Related, Health Related, etc.
- You need a data matrix with a target variable for developing predictive models.

Introduction continues

- This chapter shows how you can use the SAS Enterprise Miner's text miner nodes to help create a data matrix, reduce its dimensions in order to create a compact version of the data matrix, and include it in a SAS data set which can be used for statistical analysis and predictive modeling.
- In order to better understand the purpose of quantifying textual data, let us take an example from marketing. Suppose an automobile seller (advertiser) wants to know if a visitor to the Web is intending to buy an automobile in the near future.
- A person who intends to buy an automobile in the near future can be called an "auto intender".
- In order to determine whether a visitor to the Web is an auto intender or not, the auto seller needs to determine if the pages the visitor views frequently are auto related. The auto seller can use a predictive model that can help him decide if a web page is auto related.
- Logistic regressions, neural networks, decision trees, and support vector machines can be used for assigning class labels such as Auto Related and Not

Quantifying Textual Data: A Simplified Example

- The need for quantifying textual data also arises in query processing by search engines.
- Suppose you send the query "Car dealers" to a search engine. The search engine compares your query with a collection of documents with pre-assigned labels.
- For illustration, suppose there are only three documents in the collection (in the real world there may be thousands of documents) where each document is represented by a vector of terms as shown in Table 9.1

Table 9.1

	D1	D2	D3
Term	Document1	Document 2	Document 3
bank	0	0	3
deposit	0	0	3
report	1	1	1
taxes	0	0	3
health	8	0	0
medicine	7	0	0
car	0	8	0
driver	0	6	0
gasoline	0	1	0
domestic	0	1	1
foreign	1	1	7
exchange	0	0	4
currency	0	0	5
auto	0	7	0
dealer	0	6	1

Term-Document Matrix

The rows in a term-document matrix represent the occurrence (some measure of frequency) of a term in the documents represented by the columns. The table shown in Display 9.1 is an example of a term-document matrix.

In Table 9.1, Document1 is represented by column D1, Document2 by column2, and Document3 by column 3. There are 15 terms and 3 documents. So the term-document matrix is 15x3, and the document-term matrix, which is the transpose of the term-document matrix, is 3X15. In order to compare the query with each document, we need to represent the query as a vector consistent with the document vectors.

The query is converted to the row vector $q' = (0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1)$, where 1 indicates the occurrence of a term in the query and 0 indicates the non-occurrence. In the query vector q the 7^{th} and 15^{th} elements are equal to 1, and all other elements are equal to 0 since the 7^{th} and 15^{th} elements of the term vector are "car" and "dealer". The term vector is the first column in Table 9.1 and we can write it in vector notation as

T' = (*bank, deposit, report, taxes, health, medicine, car, driver, gasoline, domestic, foreign, exchange, currency, auto, dealer*).

In order to facilitate the explanations, let us represent the documents by the following vectors:

$D1' = (0\ 0\ 1\ 0\ 8\ 7\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0)$, $D2' = (0\ 0\ 1\ 0\ 0\ 0\ 8\ 6\ 1\ 1\ 1\ 0\ 0\ 7\ 6)$ and $D3' = (3\ 3\ 1\ 3\ 0\ 0\ 0\ 0\ 0\ 1\ 7\ 4\ 5\ 0\ 1)$.

In order to determine which document is closest to the query, we need to calculate the *scores* of each document vector with the query. The score of a document can be calculated as the inner product of the query vector and the document vector.

The score for Document 1 is: $D1'q = 0$

The score for Document 2 is $D2'q = 14$

The score for Document 3 is: $D3'q = 1$

Boolean Retrieval

The above calculations show how a query can be processed or information is retrieved from a collection of documents. An alternative way of processing queries is the Boolean Retrieval method. In this method, the occurrence of a word is represented by 1, and the non-occurrence of a word is represented by 0.

$D1' = (0\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0)$, $D2' = (0\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1)$ and $D3' = (1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 1)$.

The query is represented by a vector whose components are the frequencies of the occurrences of various words. Suppose the query consists of four occurrences of the word "car" and three occurrences of the word "dealer." Then the query can be represented by the vector:

$$q' = (0\ 0\ 0\ 0\ 0\ 0\ 4\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 3)$$

The score for Document 1 is: $D1'q = 0$

The score for Document 2 is $D2'q = 7$

The score for Document 3 is: $D3'q = 3$.

Since the score is the highest for Document2, the person who submitted the query should be directed to Document2, which contains information on auto dealers.

Document-Term Matrix and the Data Matrix

A document-term matrix is the transpose of the term-document matrix, such as the one shown in Display 9.1. A data matrix is same as the document-term matrix. The data matrix can be augmented by an additional column that shows the labels given to the documents.

In the illustration given above, the columns of the term-document matrix are the vectors $D1$, $D2$ and $D3$. The rows of the document-term or the data matrix are same as the columns of the term-document matrix.

You can verify that the vectors $D1, D2$ and D3 are the second, third and fourth columns in Table 9.1. From Table 9.1, we can define the data matrix as:

$$D = \begin{bmatrix} 0 & 0 & 1 & 0 & 8 & 7 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 8 & 6 & 1 & 1 & 1 & 0 & 0 & 7 & 6 \\ 3 & 3 & 1 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 7 & 4 & 5 & 0 & 1 \end{bmatrix} \qquad (9.1)$$

The data matrix has one row for each document and one column for each term. Therefore, in this example, its dimension is 3x15. In a SAS data set, this data matrix has 3 rows and 15 columns or 3 observations and 15 variables, where each variable is a term.

If you already know the labels of the documents, you can add an additional column with the labels. For example if we give the label 1 to a document in which the main subject is automobiles and give the label 0 if the main subject of the document is not automobiles. Then the augmented data matrix with an additional column of labels can be used to estimate a logistic regression, which can be used to classify a new document as Auto Related or Not Auto Related.

In the example presented above, the elements of the data matrix are frequencies of occurrences of the terms in the documents. The value of the ij^{th} element of D is the frequency of j^{th} term in the i^{th} document. From the D matrix shown above in Equation 9.1, you can see that the word "health" appears 8 times in Document1.

In practice, adjusted frequencies rather than the raw frequencies are used in a D matrix. The adjusted frequency can be calculated as $tfidf(i, j) = tf(i, j) * \log\left(\frac{N}{df(j)}\right)$, where $tf(i, j)$ = frequency of j^{th} term in i^{th} document (same as f_{ij}), $df(j)$ =document frequency (number of documents containing the j^{th} term) and N = Number of documents. $tfidf(i, j)$ is a measure of the relative importance of the j^{th} term in the i^{th} document. We refer to $tfidf(i, j)$ as the adjusted frequency. There are other measures of relative importance of terms in a document.

In the examples presented in this and the next section, raw frequencies (f_{ij}) are used for illustrating the methods of quantification and dimension reduction.

The main tasks of the quantifying textual data are:

1. Constructing the term-document matrix from a collection of documents where the elements of the matrix are the frequency of occurrence of the words in each document. The elements of the term-document matrix can be $tfidf$ as described above, or some other measure. The term-document matrix can be of a very large size such as 500,000x1,000,000 representing 500,000 terms and 1,000,000 documents. The corresponding data matrix has 1,000,000 rows (documents) and 500,000 columns (terms).
   Singular Value Decomposition (SVD) is applied to derive a smaller matrix such as 100x1,000,000, where the rows of the new matrix are called SVD dimensions. These dimensions may be related to meaningful concepts.

After applying the SVD, the data matrix has 1,000,000 rows and 100 columns. The rows are documents (observations) and the columns are SVD dimensions.

The above tasks can be performed by various text mining nodes as described in this chapter.

Dimension Reduction and Latent Semantic Indexing

In contrast to the data matrix used in this example, the real-life data matrices are very large with thousands of documents and tens of thousands of terms. Also, these large data matrices tend to be sparse. *Singular Value Decomposition (SVD)*, which is demonstrated in this section, is used to create smaller and denser matrices from large sparse data matrices by replacing the terms with concepts or topics. The process of creating concepts or topics from the terms is demonstrated in this section. The data matrix with concepts or terms in the columns is much smaller than the original data matrix. In this example, the D matrix (shown earlier in this chapter) which is 3x15 is replaced by a 3x3 data matrix D^*. The rows of D^* are documents, and the columns contain concepts or topics.

Singular Value Decomposition[2] factors the term-document matrix into three matrices in the following way:

$$A = U\Sigma V', \text{ where } A \text{ is a } m \times n \quad \text{term - document matrix} \tag{9.2}$$

m = number of terms and n = number of documents

U is an $m \times k$ term-concept matrix, where $k \le r = rank(A)$. The columns of U are eigenvectors of AA' and they are called the Left Singular Vectors. In this example $k = 3$.

Σ is a diagonal matrix of size $k \times k$. The diagonal elements of Σ are singular values which are the square roots of the nonzero eigenvalues of both AA' and $A'A$. The singular values in the diagonal elements of Σ are arranged in a decreasing order (as shown in the example below).

V is a concept-document matrix of size $n \times k$. The columns of V are eigenvectors of $A'A$ and they are called the Right Singular Vectors.

Applying Equation 9.2 to the term-document matrix shown in Table 9.1, we get:

$$A = \begin{bmatrix} 0 & 0 & 3 \\ 0 & 0 & 3 \\ 1 & 1 & 1 \\ 0 & 0 & 3 \\ 8 & 0 & 0 \\ 7 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 6 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 7 \\ 0 & 0 & 4 \\ 0 & 0 & 5 \\ 0 & 7 & 0 \\ 0 & 6 & 1 \end{bmatrix}, \; U = \begin{bmatrix} -0.04498 & -0.19748 & -0.18727 \\ -0.04498 & -0.19748 & -0.18727 \\ -0.08883 & -0.10807 & 0.01974 \\ -0.04498 & -0.19748 & -0.18727 \\ -0.02700 & -0.47266 & 0.57880 \\ -0.02363 & -0.41358 & 0.50645 \\ -0.56366 & 0.13471 & 0.07855 \\ -0.42275 & 0.10103 & 0.05891 \\ -0.07046 & 0.01684 & 0.00982 \\ -0.08545 & -0.04899 & -0.05261 \\ -0.17879 & -0.50303 & -0.35481 \\ -0.05998 & -0.26331 & -0.24970 \\ -0.07497 & -0.32914 & -0.31212 \\ -0.49321 & 0.11787 & 0.06873 \\ -0.43774 & 0.03520 & -0.00351 \end{bmatrix}, \; \Sigma = \begin{bmatrix} 13.86679 & 0.00000 & 0.00000 \\ 0.00000 & 11.10600 & 0.00000 \\ 0.00000 & 0.00000 & 10.41004 \end{bmatrix} \text{ and}$$

$$V = \begin{bmatrix} -0.04680 & -0.65617 & 0.75316 \\ -0.97703 & 0.18701 & 0.10221 \\ -0.20792 & -0.73108 & -0.64985 \end{bmatrix}$$

Table 9.2

	Concept 1	Concept 2	Concept 3
Term	(U1)	(U2)	(U3)
bank	-0.04498	-0.19748	-0.18727
deposit	-0.04498	-0.19748	-0.18727
report	-0.08883	-0.10807	0.01974
taxes	-0.04498	-0.19748	-0.18727
health	-0.02700	-0.47266	0.57880
medicine	-0.02363	-0.41358	0.50645
car	-0.56366	0.13471	0.07855
driver	-0.42275	0.10103	0.05891
gasoline	-0.07046	0.01684	0.00982
domestic	-0.08545	-0.04899	-0.05261
foreign	-0.17879	-0.50303	-0.35481
exchange	-0.05998	-0.26331	-0.24970
currency	-0.07497	-0.32914	-0.31212
auto	-0.49321	0.11787	0.06873
dealer	-0.43774	0.03520	-0.00351

The columns in table 9.2 can be considered as *weights* of different terms into various concepts. These weights are analogous to factor loadings in Factor Analysis. If we consider that the vectors U1, U2, and U3 represent different concepts, we will be able to label these concepts based on the weights.

Since the terms “health” and “medicine” have higher weights than other terms in column U3, we may be able to label U3 to represent the concept “health”. Since the terms “car,” “driver,” “gasoline,” “auto,” and “dealer” have higher weight than other terms in U2, we may be able to label column U2 to represent the concept “auto related”. Labeling is not so clear in the case for column U1, but if you compare the weights of the term across different documents, you may get some clues for labeling U1 also. The weight of “bank” in column U1 is higher than in columns U2 and U3. Similarly the terms “deposit,” “taxes,” “exchange,” and “currency” have higher weights in column U1 than in other columns. So we may label column U1 as Economics or Finance.

The columns of the matrix U can be considered as points in an m -dimensional space (m =15 in our example). The values in the columns are the coordinates of the points. By representing the points in a different coordinate system, you may be able to identify the concepts more clearly. Representing the points in a different coordinate system is called *rotation*. To calculate the coordinates of a point represented by a column in the U, you multiply it by a rotation matrix. For example, let the first column of the matrix U be U1. U1 is a point in 15-dimenstional space. To represent this point (U1) in a different coordinate system, you multiply U1 by a rotation matrix R. This multiplication gives new coordinates for the point represented by vector U1. By using the same ration matrix R, new coordinates can be calculated for all the columns of the term-concept matrix U. These new coordinates may show the concepts more clearly.

No rotation is performed in our example, because we were able to identify the concepts from the U matrix directly. In SAS Text Miner the Text Topic node does rotate the matrices generated by the Singular Value Decomposition. In other words, the Text Topic node performs a rotated Singular Value Decomposition.

Reducing the Number of Columns in the Data Matrix

Making use of the columns of the U matrix as weights and applying them to the rows of the data matrix D shown in Equation 9.1, we can create a data matrix with fewer columns in the following way. In general the number of columns selected in the U matrix = $k < r = rank(A)$. In our example, $k = 3$, as stated previously.

$$D^* = D \times U \times \Sigma^{-1} \tag{9.3}$$

The original data matrix D has 3 rows and 15 columns, each column representing a term. The new data matrix D^* has the same number of rows but only 3 columns.

In our example:

$$D^* = \begin{bmatrix} -0.04680 & -0.65617 & -0.75316 \\ -0.97703 & 0.18701 & -0.10221 \\ -0.20792 & -0.73108 & 0.64985 \end{bmatrix} \tag{9.4}$$

The new data matrix has 3 rows and 3 columns.

The original data matrix and reduced data matrix is shown in Tables 9.3 and 9.4.

Table 9.3

Document	bank	deposit	report	taxes	health	medicine	car	driver	gasoline	domestic	foreign	exchange	currency	auto	dealer	Target
Document1	0	0	1	0	8	7	0	0	0	0	1	0	0	0	0	H
Document2	0	0	1	0	0	0	8	6	1	1	1	0	0	7	6	A
Document3	3	3	1	3	0	0	0	0	0	1	7	4	5	0	1	E

The data matrix with reduced dimensions appended by a target column is shown in Table 9.4.

Table 9.4

Document	Concept1	Concept2	Concept3	Target
Document1	-0.04680	-0.65617	0.75316	H
Document2	-0.97703	0.18701	0.10221	A
Document3	-0.20792	-0.73108	-0.64985	E

The variables Concept1, Concept2, and Concept 3 for the three documents in the data set shown in Table 9.4 are calculated using the Equation 9.3 and shown in a matrix form in Equation 9.4. I leave it to you to carry out the matrix multiplications shown in Equation 9.3 and arrive at the compressed data matrix shown in Equation 9.4 and hence the values of Concept1, Concept2, and Concept 3 shown in Table 9.4. The variables Concept1, Concept2, and Concept3 shown in Table 9.4 can also be called svd_1, svd_2, and svd_3.

Summary of the Steps in Quantifying Textual Information

1. Retrieve the documents from the internet or from a directory on your computer and create a SAS data set. In this data set, the rows represent the documents and the columns contain the text content of the document. Use the %TMFILTER macro or the **Text Import** node.

2. Create terms. Break the stream of characters of the documents into tokens by removing all punctuation marks, reducing words to their stems or roots, and removing stop words (articles) such as a, an, the, at, in, etc. Use the **Text Parsing** node to do these tasks.

3. Reduce the number of terms. Remove redundant or unnecessary terms by using the **Text Filter** node

4. Create a term-document matrix5 with rows that correspond to terms, columns that correspond to documents (web pages, for example), and whose entities are the adjusted frequencies of occurrences of the terms.

- Create a smaller data matrix by performing a Singular Value Decomposition of the term-document matrix

Retrieving Documents from the World Wide Web:

You can retrieve documents from the World Wide Web and create SAS data sets from them using the %TMFILTER macro or the Text Import node, which is used only in a client-server environment.

You can retrieve documents from web sites using the %TMFILTER macro, as shown in Display 9.1.

Display 9.1

```
libname tmlib "C:\TextMiner\Public\SASDATA" ;

%tmfilter( dataset=tmlib.text,
    dir=c:\TextMiner\Public\tmfdir,
    destdir=c:\TextMiner\Public\tmfdir_filtered,
    numchars=32000,
    url=http://www.constitution.org/fed/federa01.htm,
    depth=1,
    force=1)
```

The code segment shown in Display 9.1 retrieves Paper 1 of the Federalist Papers from the URL http://www.constitution.org/fed/federa01.htm. When I ran the above macro, six HTML files were created and stored in the directory C:\TextMiner\Public\tmfdir. The file called file1.html contains Federalist Paper 1. A partial view of this paper is shown in Display 9.2.

Display 9.2

The Federalist No. 1

Introduction

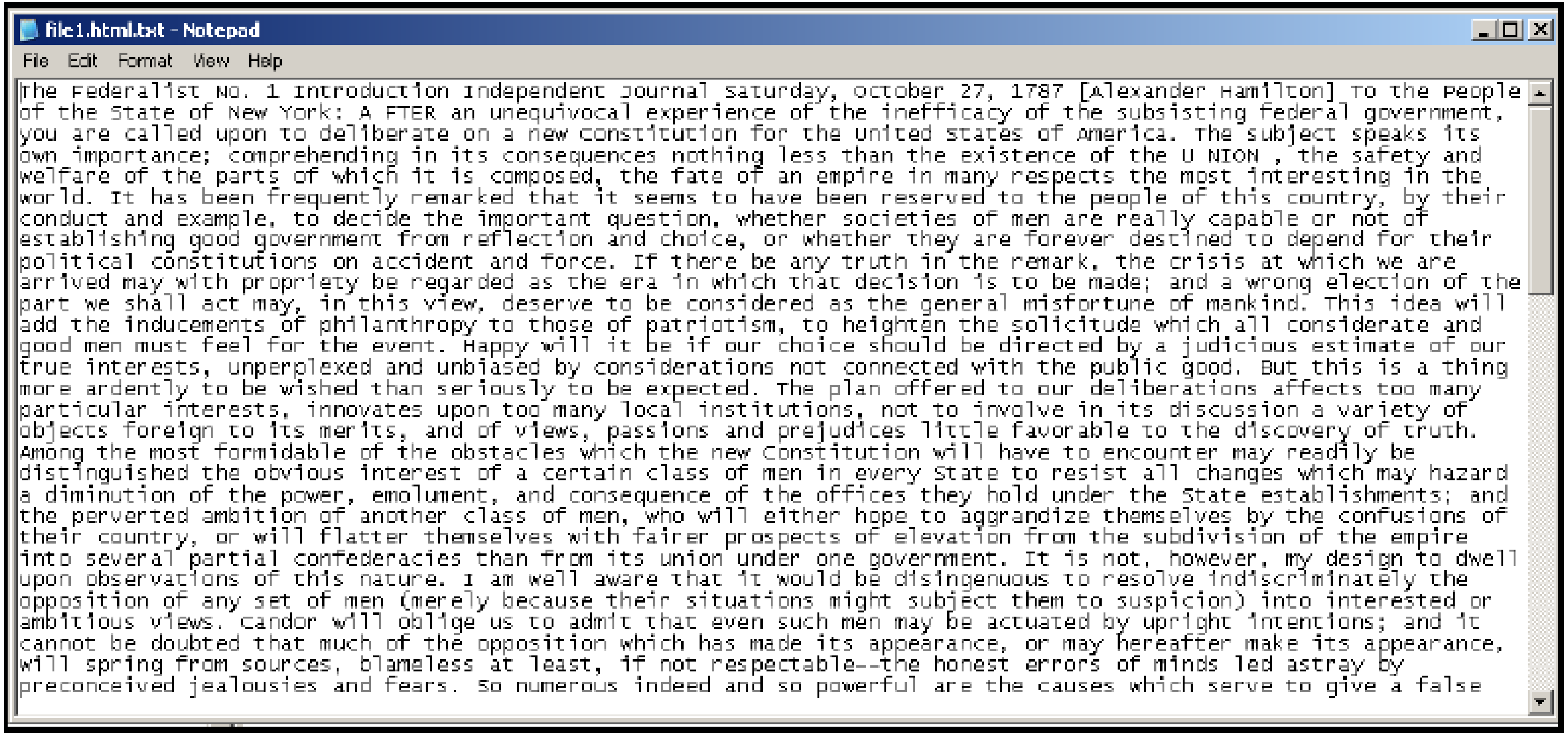
Independent Journal
Saturday, October 27, 1787
[Alexander Hamilton]

To the People of the State of New York:

AFTER an unequivocal experience of the inefficacy of the subsisting federal government, you are called upon to deliberate on a new Constitution for the United States of America. The subject speaks its own importance; comprehending in its consequences nothing less than the existence of the UNION, the safety and welfare of the parts of which it is composed, the fate of an empire in many respects the most interesting in the world. It has been frequently remarked that it seems to have been reserved to the people of this country, by their conduct and example, to decide the important question, whether societies of men are really capable or not of establishing good government from reflection and choice, or whether they are forever destined to depend for their political constitutions on accident and force. If there be any truth in the remark, the crisis at which we are arrived may with propriety be regarded as the era in which that decision is to be made; and a wrong election of the part we shall act may, in this view, deserve to be considered as the general misfortune of mankind.

The directory c:\TextMiner\Public\tmfdir_filtered contains text files with the same content as the HTML files. A partial view of the text file created is shown in Display 9.3.

Display 9.3



The Federalist No. 1 Introduction Independent Journal Saturday, October 27, 1787 [Alexander Hamilton] To the People of the State of New York: A FTER an unequivocal experience of the inefficacy of the subsisting federal government, you are called upon to deliberate on a new Constitution for the United States of America. The subject speaks its own importance; comprehending in its consequences nothing less than the existence of the U NION , the safety and welfare of the parts of which it is composed, the fate of an empire in many respects the most interesting in the world. It has been frequently remarked that it seems to have been reserved to the people of this country, by their conduct and example, to decide the important question, whether societies of men are really capable or not of establishing good government from reflection and choice, or whether they are forever destined to depend for their political constitutions on accident and force. If there be any truth in the remark, the crisis at which we are arrived may with propriety be regarded as the era in which that decision is to be made; and a wrong election of the part we shall act may, in this view, deserve to be considered as the general misfortune of mankind. This idea will add the inducements of philanthropy to those of patriotism, to heighten the solicitude which all considerate and good men must feel for the event. Happy will it be if our choice should be directed by a judicious estimate of our true interests, unperplexed and unbiased by considerations not connected with the public good. But this is a thing more ardently to be wished than seriously to be expected. The plan offered to our deliberations affects too many particular interests, innovates upon too many local institutions, not to involve in its discussion a variety of objects foreign to its merits, and of views, passions and prejudices little favorable to the discovery of truth. Among the most formidable of the obstacles which the new Constitution will have to encounter may readily be distinguished the obvious interest of a certain class of men in every State to resist all changes which may hazard a diminution of the power, emolument, and consequence of the offices they hold under the State establishments; and the perverted ambition of another class of men, who will either hope to aggrandize themselves by the confusions of their country, or will flatter themselves with fairer prospects of elevation from the subdivision of the empire into several partial confederacies than from its union under one government. It is not, however, my design to dwell upon observations of this nature. I am well aware that it would be disingenuous to resolve indiscriminately the opposition of any set of men (merely because their situations might subject them to suspicion) into interested or ambitious views. Candor will oblige us to admit that even such men may be actuated by upright intentions; and it cannot be doubted that much of the opposition which has made its appearance, or may hereafter make its appearance, will spring from sources, blameless at least, if not respectable--the honest errors of minds led astray by preconceived jealousies and fears. So numerous indeed and so powerful are the causes which serve to give a false

From the output shown in Display 9.4, you can see that six files were imported and that none of them were truncated.

Display 9.4

Frequency Percent Row Pct Col Pct	Table of TRUNCATED by OMITTED		
		OMITTED	
	TRUNCATED	0	Total
	0	6	6
		100.00	100.00
		100.00	
		100.00	
	Total	6	6
		100.00	100.00

In addition, a SAS data set named TEXT is created and stored in the directory C:\TextMiner\Public\SASDATA. This data set has six rows, each row representing a file that is imported. The first row is relevant to us, since it has Federalist Paper 1. All other files represent other information about the web site itself. Since we are interested in analyzing the Federalist Papers, we can retain only the first row. The data set consists of a variable named Text, which contains the text content of the entire Paper 1.

If you call the macro (shown in Display 9.1) 85 times using a %DO loop and append the SAS data sets created at each iteration , you get a SAS data set with 85 rows, each row representing a document. In this example, each document is a Federalist Paper.

Creating a SAS Data Set from Text Files

- If the text files are previously retrieved and stored in a directory, you can create SAS data sets from them directly. Display 9.5 shows how to create SAS data sets from text files stored in the directory C:\TextMiner\Public\CurrentText\Paper&n, where &n = 1, 2, …, 85.

Display 9.5

```
libname tmlib "C:\TextMiner\Public\SASDATA" ;

%macro sasdsn(n=,author=);
%tmfilter (dataset=Paper&n,
        dir=c:\TextMiner\Public\CurrentText\Paper&n,
        destdir=c:\TextMiner\Public\filteredText,
        ext=txt,
        language=english ,
        numchars=32000,force=1)
;
```

The arguments in the %TMFILTER macro are:

dataset =	name of the output data set.
dir =	path to the directory that contains the documents to process.
destdir =	path to specify the output in the plain text format.
ext =	extension of the files to be processed.
numchars =	length of the text variable in the output data set.
force =1	keeps the macro from terminating if the directory specified in destdir is not empty.

Display 9.5 (cont'd)

```
data Paper&n ;
 set Paper&n;
 length AUTHOR $ 16;
 length TEXT_A $ 32000;
 length Accessed_A 8 Created_A 8 Extension_A $ 32 Filtered_A $ 48
        Filteredsize_A 8 Language_A $ 7 Modified_A 8
        Name_A $ 11 omitted_A 8  Size_A 8
        Truncated_A 8 URI_A $ 60 ;
 if _N_ = 1 ;
   PAPER = &n;
   AUTHOR = "&author" ;
   TEXT_A = Text ;
   Accessed_A = Accessed;
   Created_A  = Created ;
   Extension_A = Extension ;
   Filtered_A  = Filtered ;
   Filteredsize_A = Filteredsize;
   Language_A = Language;
   Modified_A = Modified ;
   Name_A      = Name ;
   omitted_A   = Omitted ;
   Size_A      = Size ;
   Truncated_A = Truncated ;
   URI_A = URI ;
 drop text Accessed Created Extension  Filtered Filteredsize Language
      Modified Name Omitted  Size Truncated URI ;
 run ;
%if &n eq 1 %then %do;
data tmlib.Federalist;
  set Paper&n;
run;
%end ; %else %do;
proc append base=tmlib.Federalist data=Paper&n FORCE; run;
%end ;
%mend sasdsn ;
```

Using the program shown in Display 9.6, I renamed the variables and also created the target variable.

Display 9.6

```
data tmlib.Federalist2;
  length TEXT $ 32000;
  length Filtered $ 48
         Filteredsize 8 Language $ 7 Modified 8
         Name $ 11 omitted 8  Size 8
         Truncated 8 URI $ 60
         Accessed 8 Created 8 Extension $ 32 ;
  set tmlib.Federalist;
  TEXT = TEXT_A;
  drop TEXT_A;
  URI= URI_A;
  drop URI_A;
  Filtered = Filtered_A ;
  drop Filtered_A ;
  Name =  Name_A ; drop Name_A ;
  Filteredsize = Filteredsize_A ; drop Filteredsize_A;
  Language  = Language_A; drop  Language_A ;
  omitted   = omitted_A ; drop omitted_A;
  Truncated = Truncated_A ; drop Truncated_A;
  Modified = Modified_A; drop Modified_A;
  Size = Size_A; drop Size_A;
  Accessed  = Accessed_A; drop Accessed_A;
  Created = Created_A; drop Created_A;
  Extension= Extension_A; drop Extension_A;
  If author = "HAMILTON" Then TARGET = 1 ; ELSE TARGET = 0 ;
  if author in ('MADISON_HAMILTON', 'JAY') then delete;
run;
```

The %TMFILTER macro can create SAS data sets from Microsoft Word, Microsoft Excel, Adobe Acrobat, and several other types of files.

Display 9.7 shows the contents of the SAS data created by the program shown in Display 9.6.

Display 9.7

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
14	AUTHOR	Char	16
11	Accessed	Num	8
12	Created	Num	8
13	Extension	Char	32
2	Filtered	Char	48
3	Filteredsize	Num	8
4	Language	Char	7
5	Modified	Num	8
6	Name	Char	11
15	PAPER	Num	8
8	Size	Num	8
16	TARGET	Num	8
1	TEXT	Char	32000
9	Truncated	Num	8
10	URI	Char	60
7	omitted	Num	8

In the data set, there is one row per document. The variable PAPER identifies the document, and the variable TEXT contains the entire text content of the document.

The Text Import Node

The Text Import Node is an interface to the %TMFILTER macro described above.

You can use the **Text Import** node to retrieve files from the Web or from a server directory, and create data sets that can be used by other text mining nodes. The **Text Import** node relies on the SAS Document Conversion Server installed and running on a Windows machine. The machine must be accessible from the SAS Enterprise Miner Server via the host name and port number that were specified at the install time.

If you are not set up in a client-server mode, you can use the %TMFILTER macro as demonstrated in Section 9.2.1.

Creating a Data Source for Text Mining

Since the steps involved in creating a data source are the same as those demonstrated in Chapter 2, they are not shown here. Display 9.8 shows the variables in the data source.

Display 9.8

Name	Role	Level	Report	Order	Drop	Lower Limit	U
Accessed	Rejected	Interval	No		No	.	
AUTHOR	Rejected	Nominal	No		No	.	
Created	Rejected	Interval	No		No	.	
Extension	Rejected	Nominal	No		No	.	
Filtered	Rejected	Nominal	No		No	.	
Filteredsize	Rejected	Interval	No		No	.	
Language	Rejected	Nominal	No		No	.	
Modified	Rejected	Interval	No		No	.	
Name	Rejected	Nominal	No		No	.	
omitted	Rejected	Interval	No		No	.	
PAPER	ID	Interval	No		No	.	
Size	Rejected	Interval	No		No	.	
TARGET	Target	Interval	No		No	.	
TEXT	Text	Nominal	No		No	.	
Truncated	Rejected	Interval	No		No	.	
URI	Rejected	Nominal	No		No	.	

The roles of the variables TARGET, TEXT, and PAPER are set to Target, Text, and ID respectively. I set the roles of all other variables to Rejected as I do not use them in this chapter.

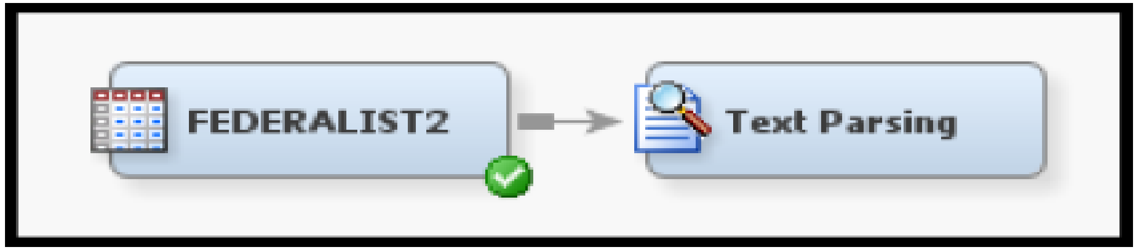
The variable TARGET takes the value 1 if the author of the paper is Hamilton and 0 otherwise, as defined by the SAS code shown in Display 9.6. The variable TEXT is the entire text of a paper.

Text Parsing Node

- Text parsing is the first step in quantifying textual data contained in a collection of documents.
- The **Text Parsing** node tokenizes the text contained in the documents by removing all punctuation marks. The character strings without spaces are called *tokens*, *words* or *terms*.
- The **Text Parsing** node also does stemming by replacing the words by their base or root. For example, the words "is" and "are" are replaced by the word "be". The **Text Parsing** node also removes stop words, which consist of mostly articles and prepositions.
- Some examples of stop words are: a, an, the, on, in, above, actually, after, and again.

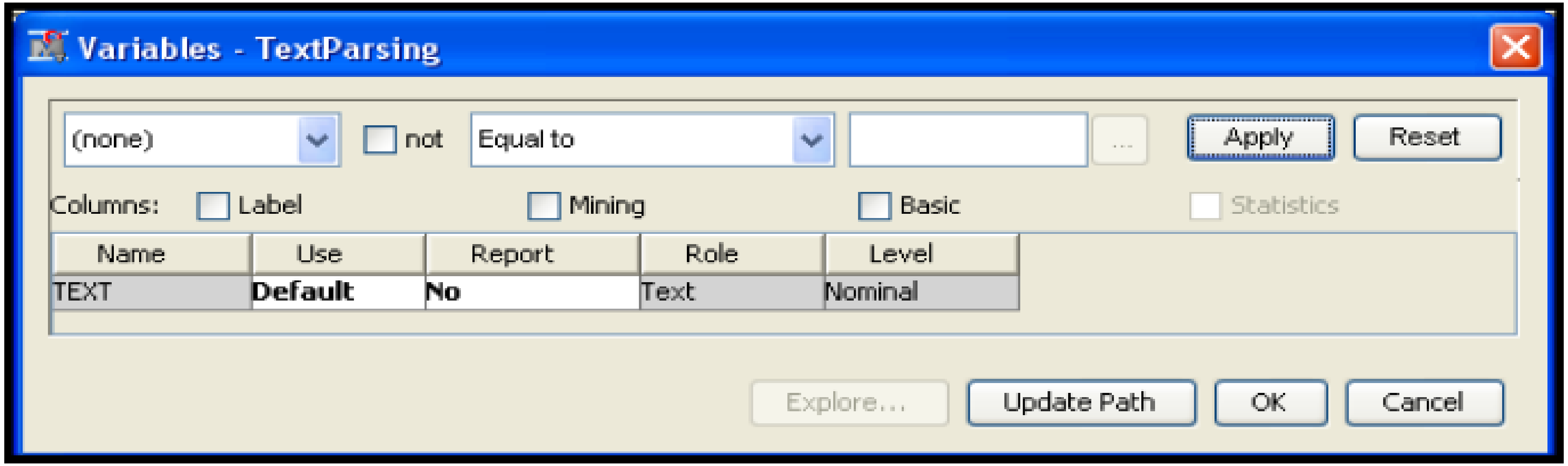
Display 9.9 shows a flow diagram with the **Text Parsing** node.

Display 9.9



Display 9.10 shows the variables processed by the **Text Parsing** node.

Display 9.10



The **Text Parsing** node processes the variable TEXT included in the analysis and collects the terms from all the documents.

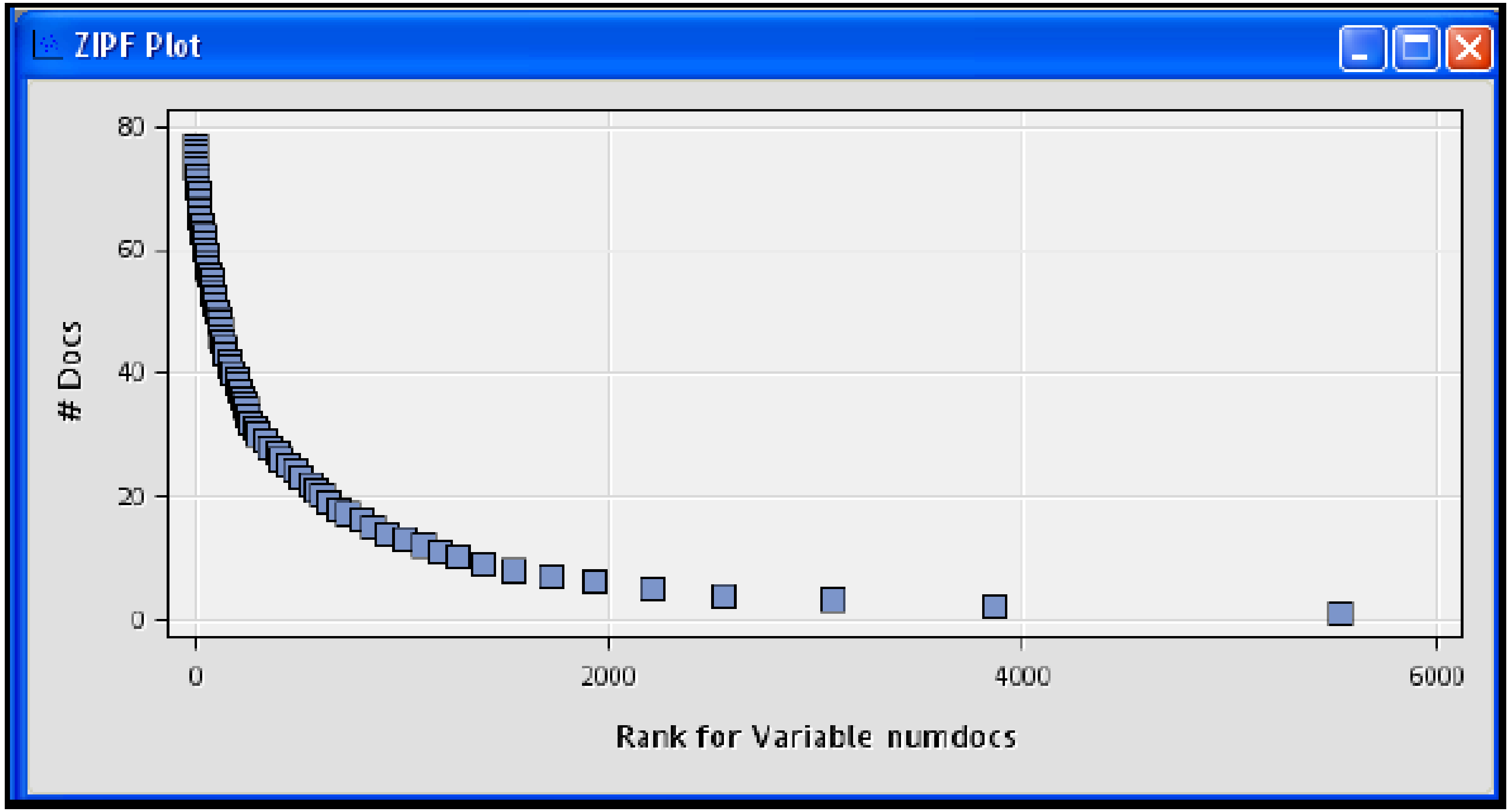
Display 9.11 shows the properties of the **Text Parsing** node.

Display 9.11

Property	Value
General	
Node ID	TextParsing
Imported Data	
Exported Data	
Notes	
Train	
Variables	
Parse	
Parse Variable	
Language	English
Detect	
Different Parts of Speech	Yes
Noun Groups	Yes
Multi-word Terms	SASHELP.ENG_MULTI
Find Entities	None
Custom Entities	
Ignore	
Ignore Parts of Speech	'Aux' 'Conj' 'Det' 'Interj' 'Part' 'Prep' 'Pron'
Ignore Types of Entities	
Ignore Types of Attributes	'Num' 'Punct'
Synonyms	
Stem Terms	Yes
Synonyms	SASHELP.ENGSYNMS
Filter	
Start List	
Stop List	SASHELP.ENGSTOP
Report	
Number of Terms to Display	20000
Status	
Create Time	2/13/13 7:03 AM

After running the **Text Parsing** node, open the Results to see a number of graphs and a table that shows the terms collected from the documents and their frequencies in the document collection. Display 9.12 shows the ZIPF plot.

Display 9.12



The horizontal axis of the ZIPF plot in Display 9.12 shows the terms by their rank and the vertical axis shows the number of documents in which the term appears. Display 9.13 gives a partial view of the rank of each term and the number of documents where the term appears.

Display 9.13

Term	Role	Attribute	Freq	# Docs	Rank for Variable numdocs	Keep	Parent/Child Status	Parent ID
+ be	...Verb	Alpha	6638	77	1	N	+	135
+ government	...Noun	Alpha	933	77	1	Y	+	75
+ have	...Verb	Alpha	1464	77	1	N	+	110
not	...Adv	Alpha	1133	77	1	N		123
+ state	... Noun	Alpha	1217	77	1	Y	+	218
+ power	... Noun	Alpha	737	76	6	Y	+	223
that	...Adv	Alpha	312	76	6	N		389
other	... Adj	Alpha	384	75	8	N		334
most	... Adv	Alpha	337	74	9	N		102
same	...Adj	Alpha	323	74	9	N		494
+ constitution	... Noun	Alpha	506	73	11	Y	+	80
+ find	...Verb	Alpha	206	73	11	Y	+	3355
+ other	... Noun	Alpha	299	73	11	N	+	488
such	...Adj	Alpha	316	73	11	N		266
any	... Adv	Alpha	209	72	15	N		136
+ great	...Adj	Alpha	340	72	15	Y	+	378
no	...Adv	Alpha	472	72	15	N		492
one	...Num	Alpha	277	72	15	N		244
+ part	... Noun	Alpha	230	72	15	N	+	150

To see term frequencies by document, you need the two data sets &EM_LIB..TEXTPARSING_TERMS and &EM_LIB..TEXTPARSING_TMOUT, where &EM_LIB refers to the directory where these data sets are stored. You can access these data sets via the **SAS Code** node.

Table 9.5 shows selected observations from the data set &EM_LIB..TEXTPARSING_TERMS.

Table 9.5

EMWS2.TEXTPARSING_TERMS

Obs	Key	Term	Role	Attribute	Freq	numdocs	_ispar	Parent	Parent_id
9	9328	abandon	Noun	Alpha	2	2		.	9328
10	2744	abandon	Verb	Alpha	3	3	.	2744	2744
11	2744	abandon	Verb	Alpha	7	6	+	.	2744
12	5058	abandoned	Verb	Alpha	3	3	.	2744	2744
13	5189	abandoning	Verb	Alpha	1	1	.	2744	2744

Table 9.6 shows the distinct values of the terms.

Table 9.6

DISTINCT TERMS

Obs	Term	Key	Role
9	abandon	2744	Verb
10	abandon	9328	Noun
11	abandoned	5058	Verb
12	abandoning	5189	Verb

Table 9.7 shows the selected observations from &EM_LIB..TEXTPARSING_TMOUT.

Table 9.7

EMWS2.TEXTPARSING_TMOUT

Obs	key	document	count
3311	2744	5	1
8146	5058	11	1
8147	5189	11	1
12813	2744	16	1
16395	5058	21	1
16971	5058	22	1
23775	9328	31	1
24908	9328	32	1
48825	2744	64	1

Table 9.8 shows a join of Tables 9.6 and 9.7.

Table 9.8

TERM_DOCUMENT MATRIX

Obs	key	document	count	Term	Role
16	2744	5	1	abandon	Verb
17	2744	16	1	abandon	Verb
18	9328	31	1	abandon	Noun
19	9328	32	1	abandon	Noun
20	2744	64	1	abandon	Verb
21	5058	11	1	abandoned	Verb
22	5058	21	1	abandoned	Verb
23	5058	22	1	abandoned	Verb
24	5189	11	1	abandoning	Verb

You can arrange the data shown in Table 9.8 into a term-document matrix. When the term "abandon" is used as a verb, it is given the key 2744, and when it is used as a noun it is given the key 9328.

Displays 9.14A and 9.14B show the SAS code that you can use to generate Tables 9.5 – 9.8. Display 9.14B shows how to create a term-document matrix using PROC TRANSPOSE. The term-document matrix shows raw frequencies of the terms by document.

Display 9.14A

```
data Textparsing_tmout;
 set &em_lib..Textparsing_tmout;
 rename _Termnum_ = key;
 rename _document_ = document;
 rename _count_ = count;

run;

 proc sql;
 create table terms as
 select distinct term, key, Role
from &em_lib..Textparsing_terms;
quit;

proc print data=&em_lib..Textparsing_terms ;
 VAR KEY TERM ROLE ATTRIBUTE FREQ NUMDOCS _ISPAR PARENT PARENT_ID;
 where term in ('abandon' 'abandoned' 'abandoning');
 title "&EM_LIB..TEXTPARSING_TERMS";
run;

proc print data=terms;
 where term in ('abandon' 'abandoned' 'abandoning');
 title " DISTINCT TERMS";
run;
```

Display 9.14B

```
proc print data=Textparsing_Tmout;
 where key in (9328,2744,5058,5189);
 title "&EM_LIB..TEXTPARSING_TMOUT";
run;
proc sql ;
 create table termdoc as
  select a.* , b.term,b.Role
  from Textparsing_tmout as a left join terms as b
  on a.key = b.key
  order by term , document ;
quit;
proc print data=termdoc ;
 title "TERM_DOCUMENT MATRIX ";
 where term in ('abandon', 'abandoned', 'abandoning');
run;
proc sort data=termdoc out=termdocs ;
by key document ;
run;

proc transpose data=termdocs out=termdocmatrix(drop=_name_) prefix=DOC;
 var count ;
 by key  ;
 id document ;
run;
```

Text Filter Node

- The **Text Filter** node reduces the number of terms by eliminating unwanted terms and filtering documents
- using a search query or a where clause. It also adjusts the raw frequencies of terms by applying term weights
- and frequency weights.

Frequency Weighting

If the frequency of i^{th} term in the j^{th} document is f_{ij}, then you can transform the raw frequencies using a formula such as $g\left(f_{ij}\right) = \log_2\left(f_{ij} + 1\right)$. Applying transformations to the raw frequencies is called *frequency weighting* since the transformation implies an implicit weighting. The function $g\left(f_{ij}\right)$ is called the fFquency Weighting Function. The weight calculated using the Frequency Weighting Function is called *frequency weight*.

Term Weighting

Suppose you want to give greater weight to terms that occur infrequently in the document collection. You can apply a weight that is inversely related to the proportion of documents that contain the term. If the proportion of documents that contain the term $t_i = P(t_i)$, then the weight applied is $w_i = \log_2\left(\frac{1}{P(t_i)}\right) + 1$, which is inverse document frequency. w_i is called the *term weight*. Another type of inverse document frequency that is used as a term weight is $w_i = \frac{1}{P(t_i)}$, where $P(t_i)$ is the proportion of documents that contain term t_i.

In the term-document matrix, the raw frequencies are replaced with adjusted frequencies calculated as the product of the term and frequency weights.

Adjusted Frequencies

Adjusted frequencies are obtained by multiplying the term weight with frequency weight.

For example, if we set $g(f_{ij}) = f_{ij}$ and $w_i = \dfrac{1}{P(t_i)}$, then the product of term frequency weight and term weight is $tfidf_{ij} = f_{ij} \times \dfrac{1}{P(t_i)}$. In general, the adjusted frequency $a_{ij} = w_i \times g(f_{ij})$. In the term-document matrix and the data matrix created by the **Text Filter** node, the raw frequencies f_{ij} are replaced by the adjusted frequencies a_{ij}.

For a list of Frequency Weighting methods (frequency weighting functions) and Term Weighting methods (term weight formulae), you can refer online to *SAS Enterprise Miner: Reference Help*. These formulae are reproduced below.

Frequency Weighting Methods

- Binary

$$g(f_{ij}) = 1 \text{ if } j^{th} \text{ term appears in the } i^{th} \text{ document,}$$
$$= 0 \text{ otherwise.}$$

- **Log** (default)

$$g(f_{ij}) = \log_2(f_{ij} + 1)$$

- Raw frequencies

$$g(f_{ij}) = f_{ij}$$

Term Weighting Methods

- Entropy

$$w_i = 1 + \sum_{j=1}^{n} \frac{\left(f_{ij} / g_i\right).\log_2\left(f_{ij} / g_i\right)}{\log_2(n)}$$, where

f_{ij} = The number of times the i^{th} term appears in the j^{th} document

g_i = The number of times the i^{th} term appears in the document collection

n = The number of documents in the collection

w_i = The weight of the i^{th} term

- Inverse Document Frequency (idf)

$$w_i = \log_2\left(\frac{1}{P(t_i)}\right) + 1, \text{ where}$$

$P(t_i) =$ Proportion of documents that contain term t_i

- Mutual Information

$$w_i = \max(C_K)\left[\log\left(\frac{P(t_i, C_k)}{P(t_i)P(C_k)}\right)\right], \text{ where}$$

$P(t_i) =$ The proportion of documents that contain term t_i

$P(C_k) =$ The proportion of documents that belong to category C_k

$P(t_i, C_k) =$ The proportion of documents that contain term t_i and belong to category C_k

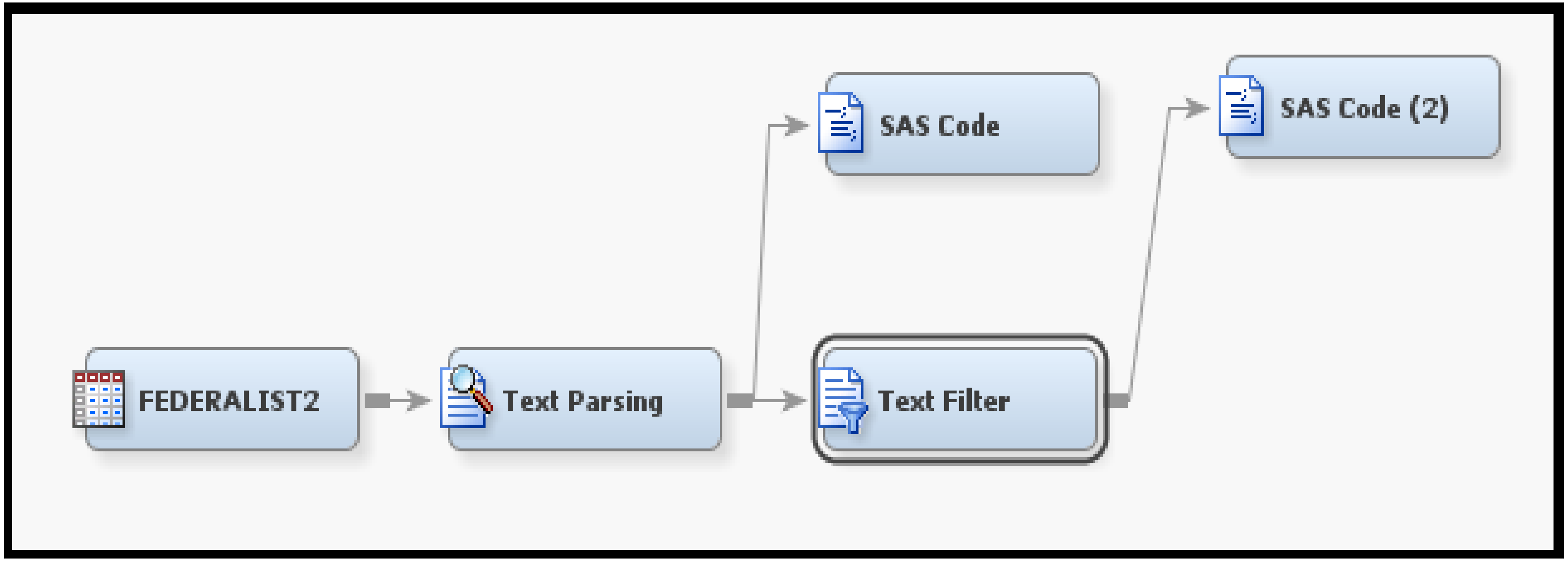
If your target is binary, then $k = 2$, $C_1 = '0'$ and $C_2 = '1'$.

- None

$w_i = 1$

Display 9.15 shows a flow diagram with the **Text Filtering** node attached to the **Text Parsing** node.

Display 9.15



Display 9.16 shows the property settings of the **Text Filter** node.

Display 9.16

Property	Value
General	
Node ID	TextFilter
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
Spelling	
Check Spelling	No
Dictionary	...
Weightings	
Frequency Weighting	None
Term Weight	Inverse Document Frequency
Term Filters	
Minimum Number of Documents	4
Maximum Number of Terms	.
Import Synonyms	...
Document Filters	
Search Expression	
Subset Documents	...
Results	
Filter Viewer	...
Spell-Checking Results	...
Exported Synonyms	...
Report	
Terms to View	All
Number of Terms to Display	20000
Status	
Create Time	2/15/13 6:04 AM

Table 9.9 gives a partial view of the terms retained by the Text Filter node .

Table 9.9

EMWS6.TEXTFILTER_TERMS

Obs	KEY	Term	Role	rolestring	Attribute	attrstring	WEIGHT	FREQ	NUMDOCS	KEEP	_ISPAR	PARENT	PARENT_ID
1	2744	abandon	Verb	Verb	Alpha	Alpha	4.68182	7	6	Y	+	.	2744
2	2744	abandon	Verb	Verb	Alpha	Alpha	0.00000	3	3	Y	.	2744	2744
3	5058	abandoned	Verb	Verb	Alpha	Alpha	0.00000	3	3	Y	.	2744	2744
4	5189	abandoning	Verb	Verb	Alpha	Alpha	0.00000	1	1	Y	.	2744	2744

Comparing Table 9.9 with 9.5, you can see that the term "abandon" with the role of a noun is dropped by the **Text Filter** node. From Table 9.10, you can see that the value of the variable KEEP for the noun "abandon" is N. Also the weight assigned to the noun "abandon" is 0, as shown in Table 9.10.

Table 9.10

EMWS6.TEXTFILTER_TERMS_TMF

Obs	KEY	Term	Role	rolestring	Attribute	attrstring	WEIGHT	FREQ	NUMDOCS	KEEP	_ISPAR	PARENT	PARENT_ID
1	2744	abandon	Verb	Verb	Alpha	Alpha	4.68182	7	6	Y	+	.	2744
2	2744	abandon	Verb	Verb	Alpha	Alpha	0.00000	3	3	Y	.	2744	2744
3	5058	abandoned	Verb	Verb	Alpha	Alpha	0.00000	3	3	Y	.	2744	2744
4	5189	abandoning	Verb	Verb	Alpha	Alpha	0.00000	1	1	Y	.	2744	2744
5	9328	abandon	Noun	Noun	Alpha	Alpha	0.00000	2	2	N		.	9328

Table 9.11 shows the terms retained prior to replacing the terms “abandoned” and “abandoning” by the root verb “abandon”.

Table 9.11

TEXT FILTER: DISTINCT TERMS (EMWS6.TEXTFILTER_TERMS)

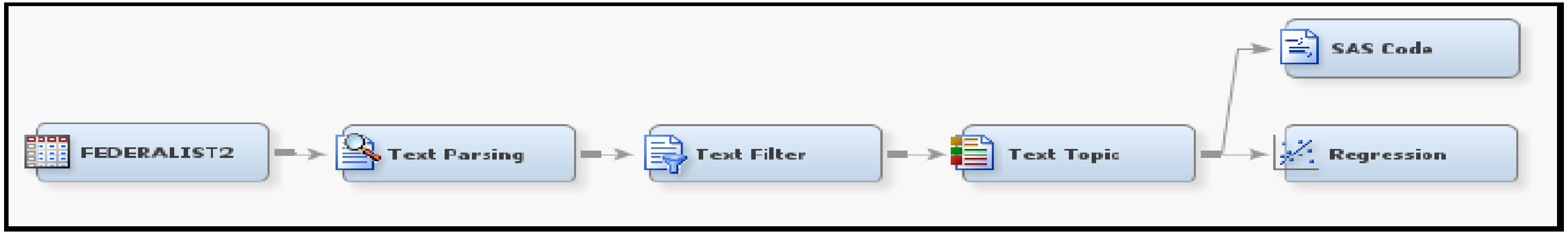
Obs	Term	KEY	Role
2	abandon	2744	Verb
3	abandoned	5058	Verb
4	abandoning	5189	Verb

Text Topic Node

The **Text Topic** node creates topics from the terms collected from the document collection done by the **Text Parsing** and **Text Filter** nodes. Different topics are created from different combinations of terms. A Singular Value Decomposition of the term-document matrix is used in creating topics from the terms. Creation of topics from the terms is similar to the derivation of concepts from the term-document matrix using Singular Value Decomposition, demonstrated in Section 9.1.2. Although the procedure used by the **Text Topic** node may be a lot more complex than the simple illustration I presented in section 9.1.2, the illustration gives a general idea of how the **Text Topic** node derives topics from the term-document matrix. I recommend that you review Sections 9.1.1 and 9.1.2, with special attention to Equations 9.1 – 9.4, matrices A, U, Σ, V, D and D^*, and Tables 9.1 – 9.4. In the illustrations in Section 9.1.2, I used the term "concept" instead of "topic." I hope that this switch of terminology does not hamper your understanding of how SVD is used to derive the topics.

Display 9.18 shows the process flow diagram with the **Text Topic** node connected to the **Text Filter** node.

Display 9.18



The property settings of the **Text Topic** node are shown in Display 9.19.

Display 9.19

Property	Value
General	
Node ID	TextTopic
Imported Data	...
Exported Data	...
Notes	...
Train	
Variables	...
User Topics	...
Term Topics	
Number of Single-term Topics	0
Learned Topics	
Number of Multi-term Topics	25
Correlated Topics	No
Results	
Topic Viewer	...
Status	
Create Time	2/17/13 9:21 AM

The term-document matrix has 2858 rows (terms) and 77 columns (documents). Selected elements of the -document-term matrix are shown in Table 9.14.

Table 9.14

Selected Elements of the Document-Term Matrix (texttopic_tmout_normalized)

Obs	_DOCUMENT_	_TERMNUM_	_count_
1	1	3	0.019791
2	5	3	0.017645
3	11	3	0.032581
4	12	3	0.018438
5	14	3	0.017255
6	16	3	0.092146
7	17	3	0.017483
8	18	3	0.017907
9	19	3	0.016800
10	20	3	0.040328
11	21	3	0.019044
12	22	3	0.078707
13	23	3	0.018257
14	24	3	0.056398
15	25	3	0.016915
16	26	3	0.052738
17	29	3	0.078976
18	31	3	0.030899
19	33	3	0.014448
20	34	3	0.014105

Tables 9.14A show s the rows corresponding to the term “abandon” (_termnum_ 2744) in the document-term matrix shown in Table 9.14. The term “abandon” is also present in tables 9.5 – 9.11 and 9.13.

Table 9.14A

The Term 'abandon' in the Document-Term Matrix (texttopic_tmout_normalized)

DOCUMENT	_TERMNUM_	_count_
5	2744	0.038649
11	2744	0.071362
16	2744	0.040366
21	2744	0.041713
22	2744	0.034479
64	2744	0.037896

The **Text Topic** node derives topics from the term-document matrix (texttopic_tmout_normalized shown in Table 9.14) using Singular Value Decomposition as outlined in Section 9.1.2. In the example presented in this section, 25 topics are derived from the 2858 terms. The topics are shown in Table 9.15.

Table 9.15

Topics and Cutoff Values				
_displayCat	_topicid	_docCutoff	_termCutoff	_name
Multiple	1	0.533	0.041	+court,+jurisdiction,+tribunal,+court,supreme
Multiple	2	0.479	0.039	+department,executive,legislative department,legislative,+judiciary department
Multiple	3	0.448	0.041	+army,military,+stand army,peace,standing
Multiple	4	0.517	0.040	+state,+power,+law,+clause,+article
Multiple	5	0.479	0.040	+government,+state government,state,people,federal
Multiple	6	0.413	0.036	+jury,+trial,+court,+case,admiralty
Multiple	7	0.424	0.038	+interest,+faction,+majority,+government,+republic
Multiple	8	0.400	0.038	+taxation,+revenue,+tax,+merchant,+state
Multiple	9	0.423	0.037	+representative,people,+state,+number,+house
Multiple	10	0.436	0.036	+bill,+state,+constitution,+right,+clause
Multiple	11	0.426	0.036	+government,+convention,congress,+state,+power
Multiple	12	0.383	0.035	+executive,plurality,+council,responsibility,+punishment
Multiple	13	0.405	0.035	+election,knowledge,+year,+state,+period
Multiple	14	0.367	0.033	+governor,president,+king,york,+state
Multiple	15	0.407	0.034	+state,+government,+majority,federal,+authority
Multiple	16	0.356	0.034	+senate,+treaty,+impeachment,+majority,+make treaty
Multiple	17	0.325	0.034	+state,+war,+republic,+nation,+confederacy
Multiple	18	0.298	0.033	+trade,commerce,+market,navigation,+state
Multiple	19	0.304	0.032	+senate,+vacancy,+appointment,+clause,president
Multiple	20	0.301	0.032	+man,+exclusion,president,+station,+office
Multiple	21	0.331	0.032	+court,+judge,legislative,judicial,judiciary
Multiple	22	0.326	0.032	+election,+senate,+state,national,+elector
Multiple	23	0.333	0.031	+state,+slave,+property,+representation,+inhabitant
Multiple	24	0.292	0.031	militia,+army,military,+state,+government
Multiple	25	0.296	0.031	+state,+convention,+government,+difficulty,+amendment

Developing a Predictive Equation Using the Output Data Set Created by the Text Topic Node

By connecting a **Regression** node to the **Text Topic** node as shown in Display 9.18, you can develop an equation that you can use for predicting who the author of a paper is.

Although I have not done it in this example, in general you should partition the data into Train, Validate, and Test data sets. Since the number of observations is small (77) in the example data set, I used the entire data set for Training only.

In the **Regression** node, I set the **Selection Model** property to Stepwise and **Selection Criterion** to None. With these settings, the model from the final iteration of the Stepwise process is selected. The selected equation is shown in Display 9.21.

Display 9.21

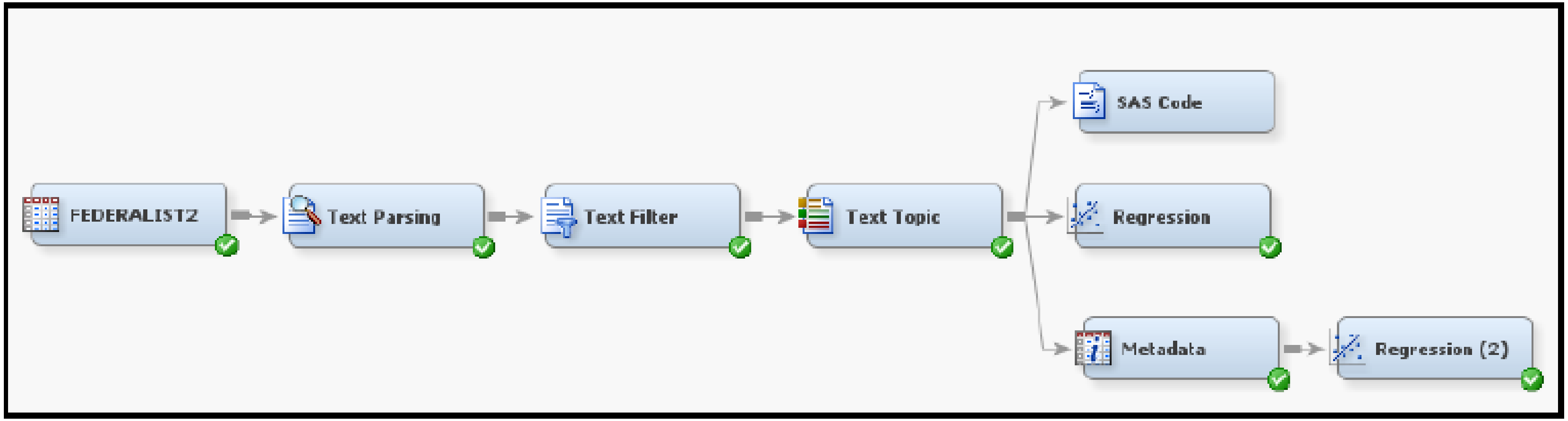
Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq	Standardized Estimate	Exp(Est)
Intercept	1	9.0857	2.7705	10.75	0.0010		999.000
TextTopic_raw10	1	8.1794	3.9221	4.35	0.0370	0.6447	999.000
TextTopic_raw2	1	-15.3612	5.1420	8.92	0.0028	-2.0465	0.000
TextTopic_raw20	1	18.9250	7.6168	6.17	0.0130	1.1384	999.000
TextTopic_raw25	1	-11.3430	4.5910	6.10	0.0135	-0.5844	0.000
TextTopic_raw9	1	-29.1967	8.0413	13.18	0.0003	-2.3848	0.000

From Display 9.21, you can see that those documents that score high on Topics 10 and 20 are more likely to be authored by Alexander Hamilton.

- In order to test the equation shown in Display 9.21, you need a Test data set. As mentioned earlier, we used the entire data set for Training only since there are not enough observations.
- Note that in the logistic regression shown in Display 9.21, only the raw scores are used. We ran an alternative equation using the indicator variables. To make the indicator variables available, we attached the **Metadata** node to the **TextTopic** node and attached a **Regression** node to the **Metadata** node, as shown in Display 9.22. We set the role of the indicator variables to Input in the **Metadata** node.

Display 9.22



The logistic regression estimated using the indicator variables is shown in Display 9.23.

Display 9.23

Analysis of Maximum Likelihood Estimates

Parameter		DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq	Exp(Est)
Intercept		1	-1.8755	0.7567	6.14	0.0132	0.153
TextTopic_11	0	1	1.6800	0.5576	9.08	0.0026	5.365
TextTopic_9	0	1	1.6030	0.5628	8.11	0.0044	4.968

Hierarchical Clustering

If you set the **Cluster Algorithm** property to Hierarchical Clustering, the **Text Cluster** nodes uses Ward's method[6] for creating the clusters. In this method, the observations (documents) are progressively combined into clusters in such a way that an objective function is minimized at each combination step. The error sum of squares for the k^{th} cluster is:

$$E_k = \sum_{i=1}^{m_k} \sum_{j=1}^{p} \left(x_{ijk} - \overline{x}_{jk} \right)^2 \qquad (9.5)$$

Where:

E_k = Error sum of squares for the k^{th} cluster

x_{ijk} = Value of the j^{th} variable at the i^{th} observation in the k^{th} cluster

$\overline{x}_{jk}$ = Mean of the j^{th} variable in the k^{th} cluster

m_k = Number of observations (documents) in the k^{th} cluster

p = Number of variables used for clustering

The objective function is the combined Error Sums of Squares given by:

$$E = \sum_{k=1}^{K} E_k \qquad (9.6)$$

Where:

$K =$ Number of clusters

At each step, the union of every possible pair of clusters is considered, and the two clusters whose union results in the minimum increase in the error sum of squares (E) are combined.

If the clusters u and v are combined to form a single cluster w, then the increase in error resulting from the union of clusters u and v is given by:

$$\Delta E_{uv} = E_w - \left(E_u + E_v\right) \tag{9.7}$$

Where:

ΔE_{uv} = Increase in the Error sum of squares due to combining clusters u and v

E_w = Error sum of squares for cluster w, which is the union of clusters u and v

E_u = Error sum of squares for cluster u

E_v = Error sum of squares for cluster v

Initially, each observation (document) is regarded as a cluster, and the observations are progressively combined into clusters as described above.

Using the Text Cluster Node

Display 9.25 shows the process diagram with two instances of the **Text Cluster** node. In the first instance, I generated clusters using the Hierarchical Clustering algorithm, and in the second instance I generated clusters using the Expectation-Maximization Clustering algorithm.

Display 9.25