# Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shared Concepts</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>The BOOLRULE Procedure</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>The TEXTMINE Procedure</td>
<td>37</td>
</tr>
<tr>
<td>4</td>
<td>The TMSCORE Procedure</td>
<td>83</td>
</tr>
</tbody>
</table>

**Subject Index**

<table>
<thead>
<tr>
<th></th>
<th>91</th>
</tr>
</thead>
</table>

**Syntax Index**

<table>
<thead>
<tr>
<th></th>
<th>93</th>
</tr>
</thead>
</table>
Chapter 1
Shared Concepts

Contents

| Introduction to Shared Concepts | 1 |
| Using CAS Sessions and CAS Engine Librefs | 1 |
| Loading a SAS Data Set onto a CAS Server | 2 |
| Details for SAS Visual Analytics Procedures | 3 |
| Multithreading | 3 |

Introduction to Shared Concepts

This book describes SAS Visual Text Analytics procedures that run on SAS Viya. One component of SAS Viya is SAS Cloud Analytic Services (CAS), which is the analytic server and associated cloud services. The following subsections describe how to set up and use CAS sessions.

The section “Details for SAS Visual Analytics Procedures” on page 3 provides details that are common to some of the procedures in this book.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```sas
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:
cas mysess;
libname mycas cas sessref=mysess;

The CAS statement creates the CAS session named mysess, and the LIBNAME statement creates the mycas CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the mysess session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

    cas mysess terminate;

For more information about the CAS statement and the LIBNAME statement, see *SAS Cloud Analytic Services: User’s Guide*. For general information about CAS and CAS sessions, see *SAS Cloud Analytic Services: Fundamentals*.

---

### Loading a SAS Data Set onto a CAS Server

Procedures in this book require the input data to reside on a CAS server. To work with a SAS data set, you must first load the data set onto the CAS server. Data loaded on the CAS server are called data tables. This section lists three methods of loading a SAS data set onto a CAS server. In this section, mycas is the name of the caslib that is connected to the mysess CAS session.

- You can use a single DATA step to create a data table on the CAS server as follows:

  ```sas
  data mycas.Sample;
    input y x @@;
  datalines;
   .46 1 .47 2 .57 3 .61 4 .62 5 .68 6 .69 7
  ;
  ```

  Note that DATA step operations might not work as intended when you perform them on the CAS server instead of the SAS client.

- You can create a SAS data set first, and when it contains exactly what you want, you can use another DATA step to load it onto the CAS server as follows:

  ```sas
  data Sample;
    input y x @@;
  datalines;
   .46 1 .47 2 .57 3 .61 4 .62 5 .68 6 .69 7 .78 8
  ;
  data mycas.Sample;
  set Sample;
  run;
  ```

- You can use the CASUTIL procedure as follows:
**Details for SAS Visual Analytics Procedures**

**Multithreading**

Threading refers to the organization of computational work into multiple tasks (processing units that can be scheduled by the operating system). A task is associated with a thread. Multithreading refers to the concurrent execution of threads. When multithreading is possible, substantial performance gains can be realized compared to sequential (single-threaded) execution. The number of threads spawned by a procedure in this book is determined by your installation.

The tasks that are multithreaded by procedures in this book are primarily defined by dividing the data that are processed on a single machine among the threads—that is, the procedures implement multithreading through a data-parallel model. For example, if the input data table has 1,000 observations and the procedure is running on four threads, then 250 observations are associated with each thread. All operations that require access to the data are then multithreaded. These operations include the following (not all operations are required for all procedures):

- variable levelization
- effect levelization
- formation of the initial crossproducts matrix
Chapter 1: Shared Concepts

- formation of approximate Hessian matrices for candidate evaluation during model selection
- objective function calculation
- gradient calculation
- Hessian calculation
- scoring of observations

In addition, operations on matrices such as sweeps can be multithreaded provided that the matrices are of sufficient size to realize performance benefits from managing multiple threads for the particular matrix operation.
# Chapter 2
## The BOOLRULE Procedure

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: BOOLRULE Procedure</td>
<td>6</td>
</tr>
<tr>
<td>PROC BOOLRULE Features</td>
<td>6</td>
</tr>
<tr>
<td>Using CAS Sessions and CAS Engine Librefs</td>
<td>7</td>
</tr>
<tr>
<td>Getting Started: BOOLRULE Procedure</td>
<td>8</td>
</tr>
<tr>
<td>Syntax: BOOLRULE Procedure</td>
<td>11</td>
</tr>
<tr>
<td>PROC BOOLRULE Statement</td>
<td>12</td>
</tr>
<tr>
<td>DOCINFO Statement</td>
<td>14</td>
</tr>
<tr>
<td>OUTPUT Statement</td>
<td>15</td>
</tr>
<tr>
<td>SCORE Statement</td>
<td>16</td>
</tr>
<tr>
<td>TERMINFO Statement</td>
<td>17</td>
</tr>
<tr>
<td>Details: BOOLRULE Procedure</td>
<td>17</td>
</tr>
<tr>
<td>BOOLLEAR for Boolean Rule Extraction</td>
<td>17</td>
</tr>
<tr>
<td>Term Ensemble Process</td>
<td>18</td>
</tr>
<tr>
<td>Rule Ensemble Process</td>
<td>20</td>
</tr>
<tr>
<td>Measurements Used in BOOLLEAR</td>
<td>21</td>
</tr>
<tr>
<td>Precision, Recall, and the F1 Score</td>
<td>21</td>
</tr>
<tr>
<td>g-Score</td>
<td>21</td>
</tr>
<tr>
<td>Estimated Precision</td>
<td>22</td>
</tr>
<tr>
<td>Improvability Test</td>
<td>22</td>
</tr>
<tr>
<td>Shrinking the Search Space</td>
<td>23</td>
</tr>
<tr>
<td>Feature Selection</td>
<td>23</td>
</tr>
<tr>
<td>Significance Testing</td>
<td>23</td>
</tr>
<tr>
<td>k-Best Search</td>
<td>24</td>
</tr>
<tr>
<td>Improvability Test</td>
<td>24</td>
</tr>
<tr>
<td>Early Stop Based on the F1 Score</td>
<td>24</td>
</tr>
<tr>
<td>Output Data Sets</td>
<td>24</td>
</tr>
<tr>
<td>CANDIDATETERMS= Data Table</td>
<td>24</td>
</tr>
<tr>
<td>RULES= Data Table</td>
<td>25</td>
</tr>
<tr>
<td>RULETERMS= Data Table</td>
<td>26</td>
</tr>
<tr>
<td>Scoring Data Set</td>
<td>26</td>
</tr>
<tr>
<td>OUTMATCH= Data Table</td>
<td>26</td>
</tr>
<tr>
<td>Examples: BOOLRULE Procedure</td>
<td>27</td>
</tr>
<tr>
<td>Example 2.1: Rule Extraction for Binary Targets</td>
<td>27</td>
</tr>
<tr>
<td>Example 2.2: Rule Extraction for a Multiclass Target</td>
<td>29</td>
</tr>
<tr>
<td>Example 2.3: Using Events in Rule Extraction</td>
<td>30</td>
</tr>
<tr>
<td>Example 2.4: Scoring</td>
<td>32</td>
</tr>
<tr>
<td>References</td>
<td>35</td>
</tr>
</tbody>
</table>
Overview: BOOLRULE Procedure

The BOOLRULE procedure is a SAS Viya procedure that enables you to extract Boolean rules from large-scale transactional data.

The BOOLRULE procedure can automatically generate a set of Boolean rules by analyzing a text corpus that has been processed by the TEXTMINE procedure and is represented in a transactional format. For example, the following rule set is generated for documents that are related to bank interest:

\[(\text{cut} \wedge \text{rate} \wedge \text{bank} \wedge \text{percent} \wedge \neg \text{sell}) \text{ or } (\text{market} \wedge \text{money} \wedge \neg \text{year} \wedge \text{percent} \wedge \neg \text{sale}) \text{ or } (\text{repurchase} \wedge \text{fee}) \text{ or } (\text{rate} \wedge \text{prime rate}) \text{ or } (\text{federal} \wedge \text{rate} \wedge \text{maturity})\]

In this example, \(^\wedge\) indicates a logical “and,” and \(\sim\) indicates a logical negation. The first line of the rule set says that if a document contains the terms “cut,” “rate,” “bank,” and “percent,” but does not contain the term “sell,” it belongs to the bank interest category.

The BOOLRULE procedure has three advantages when you use a supervised rule-based model to analyze your large-scale transactional data. First, it focuses on modeling the positive documents in a category. Therefore, it is more robust when the data are imbalanced.\(^1\) Second, the rules can be easily interpreted and modified by a human expert, enabling better human-machine interaction. Third, the procedure adopts a set of effective heuristics to significantly shrink the search space for search rules, and its basic operations are set operations, which can be implemented very efficiently. Therefore, the procedure is highly efficient and can handle very large-scale problems.

PROC BOOLRULE Features

The BOOLRULE procedure processes large-scale transactional data in parallel to achieve efficiency and scalability. The following list summarizes the basic features of PROC BOOLRULE:

- Boolean rules are automatically extracted from large-scale transactional data.
- The extracted rules can be easily understood and tuned by humans.
- Important features are identified for each category.
- Imbalanced data are handled robustly.
- Binary-class and multiclass categorization are supported.
- Events for defining labels for documents are supported.
- All processing phases use a high degree of multithreading.

\(^1\)A data table is imbalanced if it contains many more negative samples than positive samples, or vice versa.
Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```sas
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```sas
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the `mysess` session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```sas
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 1 in Chapter 1, “Shared Concepts.”
Chapter 2: The BOOLRULE Procedure

Getting Started: BOOLRULE Procedure

**NOTE:** Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections “Using CAS Sessions and CAS Engine Librefs” on page 1 and “Loading a SAS Data Set onto a CAS Server” on page 2 in Chapter 1, “Shared Concepts.”

The following DATA step creates a data table that contains 20 observations that have three variables. The **Text** variable contains the input documents. The **apple_fruit** variable contains the label of documents: a value of 1 indicates that the document is related to the apple as the fruit or to the apple tree. The **DID** variable contains the ID of the documents. Each row in the data table represents a document for analysis.

```sas
data mycas.getstart;
  infile datalines delimiter='|' missover;
  length text $150;
  input text$ apple_fruit did$;
  datalines;
  Delicious and crunchy apple is one of the popular fruits | 1 | d01
  Apple was the king of all fruits. | 1 | d02
  Custard apple or Sitaphal is a sweet pulpy fruit | 1 | d03
  apples are a common tree throughout the tropics | 1 | d04
  apple is round in shape, and tastes sweet | 1 | d05
  Tropical apple trees produce sweet apple| 1 | d06
  Fans of sweet apple adore Fuji because it is the sweetest of| 1 | d07
  this apple tree is small | 1 | d08
  Apple Store shop iPhone x and iPhone x Plus.| 0 | d09
  See a list of Apple phone numbers around the world.| 0 | d10
  Find links to user guides and contact Apple Support, | 0 | d11
  Apple counters Samsung Galaxy launch with iPhone gallery | 0 | d12
  Apple Smartphones - Verizon Wireless.| 0 | d13
  Apple mercurial chief executive, was furious.| 0 | d14
  Apple has upgraded the phone.| 0 | d15
  the great features of the new Apple iPhone x.| 0 | d16
  Apple sweet apple iphone.| 0 | d17
  Apple apple will make cars | 0 | d18
  Apple apple also makes watches| 0 | d19
  Apple apple makes computers too| 0 | d20
  ;
run;
```

These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

The following statements use the TEXTMINE procedure to parse the input text data. The generated term-by-document matrix is stored in a data table named mycas.bow. The summary information about the terms in the document collection is stored in a data table named mycas.terms.

```sas
proc textmine data=mycas.getstart language="english";
  doc_id did;
  var text;
```
parse
  nonoungroups
  entities = none
  outparent = mycas.bow
  outterms = mycas.terms
  reducef = 1;
run;

The following statements use the BOOLRULE procedure to extract rules:

proc boolrule
  data = mycas.bow
  docid = _document_
  termid = _termnum_
  docinfo = mycas.getstart
  terminfo = mycas.terms
  minsupports = 1
  mpos = 1
  gpos = 1;
  docinfo
    id = did
    targets = (apple_fruit);
  terminfo
    id = key
    label = term;
  output
    rules = mycas.rules
    ruleterms = mycas.ruleterms;
run;

The mycas.bow and mycas.terms data sets are specified as input in the DATA= and TERMINFO= options, respectively, in the PROC BOOLRULE statement. In addition, the DOCID= and TERMID= options in the PROC BOOLRULE statement specify the columns of the mycas.bow data table that contain the document ID and term ID, respectively.

The DOCINFO statement specifies the following information about the mycas.GetStart data table:

- The ID= option specifies the column that contains the document ID. The variables in this column are matched to the document ID variable that is specified in the DOCID= option in the PROC BOOLRULE statement in order to fetch target information about documents for rule extraction.

- The TARGETS= option specifies the target variables.

The TERMINFO statement specifies the following information about the mycas.terms data table:

- The ID= option specifies the column that contains the term ID. The variables in this column are matched to the term ID variable that is specified in the TERMID= option in the PROC BOOLRULE statement in order to fetch information about terms for rule extraction.

- The LABEL= option specifies the column that contains the text of the terms.

The OUTPUT statement requests that the extracted rules be stored in the data table mycas.Rules.
Figure 2.1 shows the SAS log that PROC BOOLRULE generates; the log provides information about the default configurations used by the procedure, about where the procedure runs, and about the input and output files. The log shows that the mycas.rules data table contains two observations, indicating that the BOOLRULE procedure identified two rules for the apple_fruit category.

The following statements PROC PRINT to show the contents of the mycas.rules data table that the BOOLRULE procedure generates:

```
proc print data = mycas.rules;
  var target ruleid rule F1 precision recall;
run;
```

Figure 2.2 shows the output of PROC PRINT, which contains two rules. For information about the output of the RULES= option, see the section “RULES= Data Table” on page 25.

```
<table>
<thead>
<tr>
<th>Obs</th>
<th>TARGET</th>
<th>RULEID</th>
<th>RULE</th>
<th>F1</th>
<th>PRECISION</th>
<th>RECALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>apple_fruit</td>
<td>1</td>
<td>be &amp; ~apple</td>
<td>0.93333</td>
<td>1</td>
<td>0.875</td>
</tr>
<tr>
<td>2</td>
<td>apple_fruit</td>
<td>2</td>
<td>produce</td>
<td>1.00000</td>
<td>1</td>
<td>1.000</td>
</tr>
</tbody>
</table>
```

The following statements run the BOOLRULE procedure to match rules in documents and run PROC PRINT to show the results:

```
proc boolrule
  data = mycas.bow
  docid = _document_
  termid = _termnum_
  score
    ruleterms = mycas.ruleterms
    outmatch = mycas.matches;
run;
proc print data=mycas.matches;
run;
```

Figure 2.3 shows the output of PROC PRINT, the mycas.matches data table. For information about the output of the OUTMATCH= option, see the section “OUTMATCH= Data Table” on page 26.
**Syntax: BOOLRULE Procedure**

The following statements are available in the BOOLRULE procedure:

```plaintext
PROC BOOLRULE <options> ;
   DOCINFO <options> ;
   TERMINFO <options> ;
   OUTPUT <options> ;
   SCORE <options> ;
```

The following sections describe the PROC BOOLRULE statement and then describe the other statements in alphabetical order.
PROC BOOLRULE Statement

PROC BOOLRULE < options > ;

The PROC BOOLRULE statement invokes the procedure. Table 2.1 summarizes the options in the statement by function. The options are then described fully in alphabetical order.

<table>
<thead>
<tr>
<th>option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA=</td>
<td>Specifies the input data table (which must be in transactional format) for</td>
</tr>
<tr>
<td></td>
<td>rule extraction</td>
</tr>
<tr>
<td>DOCID=</td>
<td>Specifies the variable in the DATA= data table that contains the document</td>
</tr>
<tr>
<td></td>
<td>ID</td>
</tr>
<tr>
<td>DOCINFO=</td>
<td>Specifies the input data table that contains information about documents</td>
</tr>
<tr>
<td>GNEG=</td>
<td>Specifies the minimum $g$-score needed for a negative term to be considered</td>
</tr>
<tr>
<td></td>
<td>for rule extraction</td>
</tr>
<tr>
<td>GPOS=</td>
<td>Specifies the minimum $g$-score needed for a positive term or a rule to be</td>
</tr>
<tr>
<td></td>
<td>considered for rule extraction</td>
</tr>
<tr>
<td>MAXCANDIDATES=</td>
<td>Specifies the number of term candidates to be selected for each category</td>
</tr>
<tr>
<td>MAXTRIESIN=</td>
<td>Specifies the $k_{in}$ value for $k$-best search in the term ensemble</td>
</tr>
<tr>
<td></td>
<td>process for creating a rule</td>
</tr>
<tr>
<td>MAXTRIESOUT=</td>
<td>Specifies the $k_{out}$ value for $k$-best search in the rule ensemble</td>
</tr>
<tr>
<td></td>
<td>process for creating a rule set</td>
</tr>
<tr>
<td>MINSUPPORTS=</td>
<td>Specifies the minimum number of documents in which a term needs to appear</td>
</tr>
<tr>
<td></td>
<td>in order for the term to be used for creating a rule</td>
</tr>
<tr>
<td>MNEG=</td>
<td>Specifies the $m$ value for computing estimated precision for negative</td>
</tr>
<tr>
<td></td>
<td>terms</td>
</tr>
<tr>
<td>MPOS=</td>
<td>Specifies the $m$ value for computing estimated precision for positive</td>
</tr>
<tr>
<td></td>
<td>terms</td>
</tr>
<tr>
<td>TERMID=</td>
<td>Specifies the variable in the DATA= data table that contains the term ID</td>
</tr>
<tr>
<td>TERMINFO=</td>
<td>Specifies the input data table that contains information about terms</td>
</tr>
</tbody>
</table>

You must specify the following option:

DATA=CAS-libref.data-table
DOC=CAS-libref.data-table

names the input data table for PROC BOOLRULE to use. CAS-libref.data-table is a two-level name, where
**CAS-libref** refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about **CAS-libref**, see the section “Using CAS Sessions and CAS Engine Librefs” on page 7.

**data-table** specifies the name of the input data table.

Each row of the input data table must contain one variable for the document ID and one variable for the term ID. Both the document ID variable and the term ID variable can be either a numeric or character variable. The BOOLRULE procedure does not assume that the data table is sorted by either document ID or term ID.

You can also specify the following options:

**DOCID=variable**

specifies the *variable* that contains the ID of each document. The document ID can be either a number or a string of characters.

**DOCINFO=CAS-libref.data-table**

names the input data table that contains information about documents. **CAS-libref.data-table** is a two-level name, where **CAS-libref** refers to the caslib and session identifier, and **data-table** specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 7.

Each row of the input data table must contain one variable for the document ID. The BOOLRULE procedure uses the document ID in the DATA= data table to search for the document ID variable in this data table to obtain information about documents (for example, the categories of each document).

**GNEG=g-value**

specifies the minimum g-score needed for a negative term to be considered for rule extraction in the term ensemble. If you do not specify this option, the value that is specified for the GPOS= option (or its default value) is used. For more information about g-score, see the section “g-Score” on page 21.

**GPOS=g-value**

specifies the minimum g-score needed for a positive term to be considered for rule extraction in the term ensemble. A rule also needs to have a g-score that is higher than g-value to be considered in the rule ensemble. The g-value is also used in the improvability test. A rule is improvable if the g-score that is computed according to the improvability test is larger than g-value. By default, GPOS=8.

**MAXCANDIDATES=n**

**MAXCANDS=n**

specifies the number of term candidates to be selected for each category. Rules are built by using only these term candidates. By default, MAXCANDS=500.

**MAXTRIESIN=n**

specifies the $k_{in}$ value for the $k$-best search in the term ensemble process for creating rules. For more information, see the section “$k$-Best Search” on page 24. By default, MAXTRIESIN=150.
MAXTRIESOUT=n
specifies the $k_{out}$ value for the $k$-best search in the rule ensemble process for creating a rule set. For more information, see the section “$k$-Best Search” on page 24. By default, MAXTRIESOUT=50.

MINSUPPORTS=n
specifies the minimum number of documents in which a term needs to appear in order for the term to be used for creating a rule. By default, MINSUPPORTS=3.

MNEG=m
specifies the $m$ value for computing estimated precision for negative terms. If you do not specify this option, the value specified for the MPOS= option (or its default value) is used.

MPOS=m
specifies the $m$ value for computing estimated precision for positive terms. By default, MPOS=8.

TERMID=variable
specifies the variable that contains the ID of each term. The variable can be either a number or a string of characters. If the TERMINFO= option is not specified, variable is also used as the label of terms.

TERMINFO=CAS-libref.data-table
names the input data table that contains information about terms. CAS-libref.data-table is a two-level name, where CAS-libref refers to the caslib and session identifier, and data-table specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 7.

Each row of the input data table must contain one variable for the term ID. If you specify this option, you must use the TERMINFO statement to specify which variables in the data table contain the term ID and the term label, respectively. The BOOLRULE procedure uses the term ID in the DATA= data table to search for the term ID variable in this data table to obtain information about the terms. If you do not specify this option, the content of the TERMID= variable is also used as the label of terms.

DOCINFO Statement

DOCINFO < options > ;

The DOCINFO statement specifies information about the data table that is specified in the DOCINFO= option in the PROC BOOLRULE statement.

You can specify the following options:

EVENTS=(value1, value2, . . .)
specifies the values of target variables that are considered as positive events or categories of interest as follows:

- When TARGETTYPE=BINARY, the values of each target variable that is specified in the TARGET= option correspond to positive events. All other values correspond to negative events.
- When TARGETTYPE=BINARY, for any variable specified in the TARGET= option that is a numeric variable, “1” is considered to be a positive event by default.
- When TARGETTYPE=BINARY, for any variable specified in the TARGET= option that is a character variable, “Y” is considered to be a positive event by default.
You cannot specify this option when TARGETTYPE=MULTICLASS.

ID=variable

specifies the variable that contains the document ID. To fetch the target information about documents, the values in the variable are matched to the document ID variable that is specified in the DOCID= option in the PROC BOOLRULE statement. The variable can be either a numeric variable or a character variable. Its type must match the type of the variable that is specified in the DOCID= option in the PROC BOOLRULE statement.

TARGET=(variable, variable, . . .)

specifies the target variables. A target variable can be either a numeric variable or a character variable.

- When TARGETTYPE=BINARY, you can specify multiple target variables, and each target variable corresponds to a category.
- When TARGETTYPE=MULTICLASS, you can specify only one target variable, and each of its levels corresponds to a category.

TARGETTYPE=BINARY | MULTICLASS

specifies the type of the target variables. You can specify the following values:

BINARY

depicts that multiple target variables can be specified and each target variable corresponds to a category.

MULTICLASS

depicts that only one target variable can be specified and each level of the target variable corresponds to a category.

By default, TARGETTYPE=BINARY.

OUTPUT Statement

OUTPUT < options > ;

The OUTPUT statement specifies the data tables that contain the results that the BOOLRULE procedure generates.

You can specify the following options:

CANDIDATETERMS=CAS-libref.data-table

specifies a data table to contain the terms that have been selected by the BOOLRULE procedure for rule creation. CAS-libref.data-table is a two-level name, where CAS-libref refers to the caslib and session identifier, and data-table specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 7.

If MAXCANDIDATES=p in the BOOLRULE statement, the procedure selects at most p terms for each category to be considered for rule extraction. For more information about this data table, see the section “Output Data Sets” on page 24.
RULES=\
  \textit{CAS-libref.data-table}\

specifies a data table to contain the rules that have been generated by the BOOLRULE procedure for each category. \textit{CAS-libref.data-table} is a two-level name, where \textit{CAS-libref} refers to the caslib and session identifier, and \textit{data-table} specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 7.

For more information about this data table, see the section “Output Data Sets” on page 24.

RULETERMS=\
  \textit{CAS-libref.data-table}\

specifies a data table to contain the terms in each rule that is generated by the BOOLRULE procedure. \textit{CAS-libref.data-table} is a two-level name, where \textit{CAS-libref} refers to the caslib and session identifier, and \textit{data-table} specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 7.

For more information about this data table, see the section “Output Data Sets” on page 24.

**SCORE Statement**

```
SCORE <options>;
```

The SCORE statement specifies the input data table that contains the terms in rules and the output data table to contain the scoring results.

You can specify the following \textit{options}:

OUTMATCH=\
  \textit{CAS-libref.data-table}\

specifies a data table to contain the rule-matching results (that is, whether a document satisfies a rule). \textit{CAS-libref.data-table} is a two-level name, where \textit{CAS-libref} refers to the caslib and session identifier, and \textit{data-table} specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 7.

For more information about this data table, see the section “Scoring Data Set” on page 26.

RULETERMS=\
  \textit{CAS-libref.data-table}\

specifies a data table that contains the terms in each rule that the BOOLRULE procedure generates. \textit{CAS-libref.data-table} is a two-level name, where \textit{CAS-libref} refers to the caslib and session identifier, and \textit{data-table} specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 7.

For more information about this data table, see the section “RULETERMS= Data Table” on page 26.
**TERMINFO Statement**

```
TERMINFO < options > ;
```

The TERMINFO statement specifies information about the data table that is specified in the TERMINFO= option in the PROC BOOLRULE statement. If you specify the TERMINFO= data table in the PROC BOOLRULE statement, you must also include this statement to specify which variables in the data table contain the term ID and the term label, respectively.

You can specify the following options:

- **ID=**variable
  specifies the variable that contains the term ID. To fetch the text of terms, the values in variable are matched to the term ID variable that is specified in the TERMID= option in the PROC BOOLRULE statement. The variable can be either a numeric variable or a character variable. Its type must match the type of the variable that is specified in the TERMID= option in the PROC BOOLRULE statement.

- **LABEL=**variable
  specifies the variable that contains the text of the terms, where variable must be a character variable.

---

**Details: BOOLRULE Procedure**

PROC BOOLRULE implements the BOOLLEAR technique for rule extraction. This section provides details about various aspects of the BOOLRULE procedure.

**BOOLLEAR for Boolean Rule Extraction**

Rule-based text categorization algorithms use text rules to classify documents. Text rules are interpretable and can be effectively learned even when the number of positive documents is very limited. BOOLLEAR (Cox and Zhao 2014) is a novel technique for Boolean rule extraction. When you supply a text corpus that contains multiple categories, BOOLLEAR extracts a set of binary rules from each category and represents each rule in the form of a conjunction, where each item in the conjunction denotes the presence or absence of a particular term. The BOOLLEAR process is as follows (criteria and measurements that are used in this process are described in the next section):

1. Use an information gain criterion to form an ordered term candidate list. The term that best predicts the category is first on the list, and so on. Terms that do not have a significant relationship to the category are removed from this list. Set the current term to the first term.

2. Determine the “estimated precision” of the current term. The estimated precision is the projected percentage of the term’s occurrence with the category in out-of-sample data, using additive smoothing. Create a rule that consists of that term.

3. If the “estimated precision” of the current rule could not possibly be improved by adding more terms as qualifiers, then go to step 6.
4. Starting with the next term on the list, determine whether the conjunction of the current rule with that term (via either term presence or term absence) significantly improves the information gain and also improves estimated precision.

5. If there is at least one combination that meets the criterion in step 4, choose the combination that yields the best estimated precision, and go to step 3 with that combination. Otherwise, continue to step 6.

6. If the best rule obtained in step 3 has a higher estimated precision than the current “highest precision” rule, replace the current rule with the new rule.

7. Increment the current term to the next term in the ordered candidate term list and go to step 2. Continue repeating until all terms in the list have been considered.

8. Determine whether the harmonic mean of precision and recall (the F1 score) of the current rule set is improved by adding the best rule obtained by steps 1 to 7. If it is not, then exit.

9. If so, remove from the document set all documents that match the new rule, add this rule to the rule set, and go to step 1 to start creating the next rule in the rule set.

BOOLLEAR contains two essential processes for rule extraction: a term ensemble process (steps 4–5), which creates rules by adding terms; and a rule ensemble process (steps 2–9), which creates a rule set. The rule set can then be used for either content exploration or text categorization. Both the term ensemble process and the rule ensemble process are iterative processes. The term ensemble process forms an inner loop of the rule ensemble process. Efficient heuristic search strategies and sophisticated evaluation criteria are designed to ensure state-of-the-art performance of BOOLLEAR.

**Term Ensemble Process**

The term ensemble process iteratively adds terms to a rule. When the process finishes, it returns a rule that can be used as a candidate rule for the rule ensemble process. Figure 2.4 shows the flowchart of the term ensemble process.
Before adding terms to a rule, BOOLLEAR first sorts the candidate terms in descending order according to their $g$-score with respect to the target category. It then starts to add terms to the rule iteratively. In each iteration of the term ensemble process, BOOLLEAR takes a term $t$ from the ordered candidate term list and determines whether adding the term to the current rule $r$ can improve the rule’s estimated precision. To ensure that the term is good enough, BOOLLEAR tries $k_{in} - 1$ additional terms in the term list, where $k_{in}$ is the maximum number of terms to examine for improvement. If none of these terms is better (results in a lower $g$-score of the current rule $r$) than term $t$, the term is considered to be $k$-best, where $k = k_{in}$, and BOOLLEAR updates the current rule $r$ by adding term $t$ to it. If one of the $k_{in} - 1$ additional terms is better than term $t$, BOOLLEAR sets that term as $t$ and tries $k_{in} - 1$ additional terms to determine whether this new $t$ is better than all of those additional terms. BOOLLEAR repeats until the current term $t$ is $k$-best or until it reaches the end of the term list. After a term is added to the rule, BOOLLEAR marks the term as used and continues to identify the next $k$-best term from the unused terms in the sorted candidate term list. When a $k$-best term is identified, BOOLLEAR adds it to the rule. BOOLLEAR keeps adding $k$-best terms until the rule cannot be further improved. By trying to identify a $k$-best term instead of the global best, BOOLLEAR shrinks its search space to improve its efficiency.
Rule Ensemble Process

The rule ensemble process iteratively creates and adds new rules to a rule set. When the process finishes, it returns the rule set, which can then be used for text categorization. Figure 2.5 shows the flowchart of the rule ensemble process.

Figure 2.5 Rule Ensemble for Creating a Rule Set

In each iteration of the rule ensemble process, BOOLLEAR tries to find a rule r that has the highest precision in classifying the previously unclassified positive samples. For the first iteration, all samples are unclassified. To ensure that the precision of rule r is good enough, BOOLLEAR generates $k_{out} - 1$ additional rules, where $k_{out}$ is an input parameter that you specify in the MAXTRIESOUT= option in the PROC BOOLRULE statement. If one of these rules has a higher precision than rule r, BOOLLEAR sets that rule as the new rule r and generates another $k_{out} - 1$ rules to determine whether this new rule is the best among them. BOOLLEAR repeats this process until the current rule r is better than any of the $k_{out} - 1$ rules that are generated after it. The obtained rule r is called a $k$-best rule, where $k = k_{out}$. When BOOLLEAR obtains a $k$-best rule, it adds that rule to the rule set and removes from the corpus all documents that satisfy the rule. In order to reduce the possibility of generating redundant rules, BOOLLEAR then determines whether the F1 score of the rule set is improved. If the F1 score is improved, BOOLLEAR goes to the next iteration and uses the updated corpus to generate another rule. Otherwise, it treats the current rule set as unimprovable, stops the search, and outputs the currently obtained rule set. Note that to identify a “good” rule, BOOLLEAR does not go through all the potential rules to find the global “best,” because doing so can be computationally intractable when the number of candidate terms is large. Also, before BOOLLEAR generates a rule, it orders the terms in the candidate term set by their correlation to the target. So it is reasonable to expect that the obtained $k$-best rule is close to a globally best rule in terms of its capability for improving the F1 score of the rule set.

For information about the F1 score, see the section “Precision, Recall, and the F1 Score” on page 21.
Measurements Used in BOOLLEAR

This section provides detailed information about the measurements that are used in BOOLLEAR to evaluate terms and rules.

**Precision, Recall, and the F1 Score**

Precision measures the probability that the observation is actually positive when a classifier predicts it to be positive; recall measures the probability that a positive observation will be recognized; and the F1 score is the harmonic mean of precision and recall. A good classifier should be able to achieve both high precision and high recall. The precision, recall, and F1 score are defined as

\[
\text{precision} = \frac{TP}{TP + FP},
\]

\[
\text{recall} = \frac{TP}{TP + FN},
\]

\[
F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

where TP is the true-positive (the number of documents that are predicted to be positive and are actually positive), FP is the false-positive (the number of documents that are predicted to be positive but are actually negative), TN is the true-negative (the number of documents that are predicted to be negative and are actually negative), and FN is the false-negative (the number of documents that are predicted to be negative but are actually positive). A classifier thus obtains a high F1 score if and only if it can achieve both high precision and high recall. The F1 score is a better measurement than accuracy when the data are imbalanced, because a classifier can obtain very high accuracy by predicting that all samples belong to the majority category.

**g-Score**

BOOLLEAR uses the g-test (which is also known as the likelihood-ratio or maximum likelihood statistical significance test) as an information gain criterion to evaluate the correlation between terms and the target. The g-test generates a g-score, which has two beneficial properties: as a form of mutual information, it is approximately equivalent to information gain in the binary case; and because it is distributed as a chi-square, it can also be used for statistical significance testing. The g-test is designed to compare the independence of two categorical variables. Its null hypothesis is that the proportions at one variable are the same for different values of the second variable. Given the TP, FP, FN, and TN of a term, the term’s g-score can be computed as

\[
g = 2 \times \sum_{i=\{TP,TN,FP,FN\}} O(i) \log \left( \frac{O(i)}{E(i)} \right)
\]

**Note:** Accuracy is defined as \( \frac{TP + TN}{TP + FP + TN + FN} \).
Chapter 2: The BOOLRULE Procedure

\[
\begin{align*}
O(\text{TP}) &= \text{TP} \\
O(\text{FP}) &= \text{FP} \\
O(\text{TN}) &= \text{TN} \\
O(\text{FN}) &= \text{FN} \\
E(\text{TP}) &= \frac{(\text{TP} + \text{FP}) \times P}{P + N} \\
E(\text{FP}) &= \frac{(\text{TP} + \text{FP}) \times N}{P + N} \\
E(\text{TN}) &= \frac{(\text{TN} + \text{FN}) \times N}{P + N} \\
E(\text{FN}) &= \frac{(\text{TN} + \text{FN}) \times P}{P + N}
\end{align*}
\]

where \( P \) is the number of positive documents; \( N \) is the number of negative documents; \( O(\text{TP}), O(\text{FP}), O(\text{TN}), \) and \( O(\text{FN}) \) refer to the observed TP, FP, TN, and FN of a term; and \( E(\text{TP}), E(\text{FP}), E(\text{TN}), \) and \( E(\text{FN}) \) refer to the expected TP, FP, TN, and FN of a term. A term has a high \( g \)-score if it appears often in positive documents but rarely in negative documents, or vice versa.

**Estimated Precision**

Estimated precision helps BOOLLEAR shorten its search path and avoid generating overly specific rules. The precision is estimated by a form of additive smoothing with additional correction (\( \text{err}_i \)) to favor shorter rules over longer rules:

\[
\text{precision}^m_i(t) = \frac{\text{TP}_{i,t} + \frac{P}{N+P} \times m}{\text{TP}_{i,t} + \text{FP}_{i,t} + m} - \text{err}_{i-1}
\]

\[
\text{err}_i = \frac{\text{TP}_{i,t}}{\text{TP}_{i,t} + \text{FP}_{i,t}} - \frac{\text{TP}_{i,t} + \frac{P}{N+P} \times m}{\text{TP}_{i,t} + \text{FP}_{i,t} + m} + \text{err}_{i-1}
\]

In the preceding equations, \( m(\leq 1) \) is a parameter that you specify for bias correction. A large \( m \) is called for when a very large number of rules are evaluated, in order to minimize selection bias. \( \text{TP}_{i,t} \) and \( \text{FP}_{i,t} \) are the true-positive and false-positive of rule \( t \) when the length of the rule is \( i \).

**Improvability Test**

BOOLLEAR tests for improvability in the term ensemble step for “in-process” model pruning. To determine whether a rule is improvable, BOOLLEAR applies the \( g \)-test to a perfect confusion table that is defined as

\[
\begin{array}{c|c}
\text{TP} & 0 \\
0 & \text{FP}
\end{array}
\]

In this table, TP is the true-positive of the rule and FP is the false-positive of the rule. The \( g \)-score that is computed by using this table reflects the maximum \( g \)-score that a rule could possibly obtain if a perfectly
discriminating term were added to the rule. If the g-score is smaller than a number that you specify to indicate a maximum p-value for significance in the GPOS= and GNEG= options, BOOLLEAR considers the rule to be unimprovable.

---

**Shrinking the Search Space**

Exhaustively searching the space of possible rules is impractical because of the exponential number of rules that would have to be searched ($2^m$ rules, where $m$ is the number of candidate terms). In addition, an exhaustive search usually leads to overfitting by generating many overly specific rules. Therefore, BOOLLEAR implements the strategies described in the following sections to dramatically shrink the search space to improve its efficiency and help it avoid overfitting.

**Feature Selection**

BOOLLEAR uses the g-test to evaluate terms. Assume that MAXCANDIDATES=$p$ and MINSUPPORTS=$c$ in the PROC BOOLRULE statement. A term is added to the ordered candidate term list if and only if the following two conditions hold:

1. The term is a top $p$ term according to its g-score.
2. The term appears in more than $c$ documents.

The size of the candidate term list controls the size of the search space. The smaller the size, the fewer terms are used for rule extraction, and therefore the smaller the search space is.

**Significance Testing**

In many rule extraction algorithms, rules are built until they perform perfectly on a training set, and pruning is applied afterwards. In contrast, BOOLLEAR prunes “in-process.” The following three checks are a form of in-process pruning; rules are not expanded when their expansion does not meet these basic requirements. These requirements help BOOLLEAR truncate its search path and avoid generating overly specific rules.

- **Minimum positive document coverage**: BOOLLEAR requires that a rule be satisfied by at least $s$ positive documents, where $s$ is the value of the MINSUPPORTS= option in the PROC BOOLRULE statement.

- **Early stop based on g-test**: BOOLLEAR stops searching when the g-score that is calculated for improving (or starting) a rule does not meet required statistical significance levels.

- **Early stop based on estimated precision**: BOOLLEAR stops building a rule when the estimated precision of the rule does not improve when the current best term is added to the rule. This strategy helps BOOLLEAR shorten its search path.
**k-Best Search**

In the worst case, BOOLLEAR could still examine an exponential number of rules, although the heuristics described here minimize that chance. But because the terms are ordered by predictiveness of the category beforehand, a k-best search is used to further improve the efficiency of BOOLLEAR: If BOOLLEAR tries unsuccessfully to expand (or start) a rule numerous times with the a priori “best” candidates, then the search can be prematurely ended. Two optional parameters, \( k_{in} \) and \( k_{out} \), determine the maximum number of terms and rules to examine for improvement. The \( k_{in} \) parameter (which is specified in the MAXTRIESIN= option) is used in the term ensemble process: if \( k_{in} \) consecutive terms have been checked for building possible rules and none of them are superior to the best current rule, the search is terminated. The \( k_{out} \) parameter (which is specified in the MAXTRIESOUT= option) is used in the rule ensemble process: if \( k_{out} \) consecutive terms have been checked to add to a rule and they do not generate a better rule, then the search for expanding that rule is terminated. This helps BOOLLEAR shorten its search path, even with a very large number of candidate terms, with very little sacrifice in accuracy.

**Improvability Test**

BOOLLEAR tests whether adding a theoretical perfectly discriminating term to a particular rule could possibly have both a statistically significant result and a higher estimated precision than the current rule. If it cannot, then the current rule is recognized without additional testing as the best possible rule, and no further expansion is needed.

**Early Stop Based on the F1 Score**

BOOLLEAR stops building the rule set if adding the current best rule does not improve the rule set’s F1 score. Thus the F1 score is treated as the objective to maximize.

---

**Output Data Sets**

This section describes the output data sets that PROC BOOLRULE produces when you specify the corresponding option in the OUTPUT statement.

**CANDIDATETERMS= Data Table**

The CANDIDATETERMS= option in the OUTPUT statement specifies a data table to contain the terms that have been selected by the procedure for rule creation. If MAXCANDIDATES=\( p \) in the PROC BOOLRULE statement, the procedure selects a maximum of \( p \) terms for each category.

Table 2.2 shows the fields in this data table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>The category that the term is selected for (this field corresponds to the Target field in the RULES= data table)</td>
</tr>
<tr>
<td>Rank</td>
<td>The rank of the term in the ordered term list for the category (term rank starts from 1)</td>
</tr>
<tr>
<td>Term</td>
<td>A lowercase version of the term</td>
</tr>
</tbody>
</table>
Table 2.2  continued

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>The term identifier of the term</td>
</tr>
<tr>
<td>GScore</td>
<td>The g-score of the term that is obtained for the target category</td>
</tr>
<tr>
<td>Support</td>
<td>The number of documents in which the term appears</td>
</tr>
<tr>
<td>TP</td>
<td>The number of positive documents in which the term appears</td>
</tr>
<tr>
<td>FP</td>
<td>The number of negative documents in which the term appears</td>
</tr>
</tbody>
</table>

RULES= Data Table

The RULES= option in the OUTPUT statement specifies the output data table to contain the rules that have been generated for each category.

Table 2.3 shows the fields in this data table.

Table 2.3  Fields in the RULES= Data Table

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>The target category that the term is selected to model</td>
</tr>
<tr>
<td>Target_var</td>
<td>The variable that contains the target</td>
</tr>
<tr>
<td>Target_val</td>
<td>The value of the target variable</td>
</tr>
<tr>
<td>Ruleid</td>
<td>The ID of a rule (Ruleid starts from 1)</td>
</tr>
<tr>
<td>Ruleid_loc</td>
<td>The ID of a rule in a rule set (in each rule set, Ruleid_loc starts from 1)</td>
</tr>
<tr>
<td>Rule</td>
<td>The text content of the rule</td>
</tr>
<tr>
<td>TP</td>
<td>The number of positive documents that are satisfied by the rule set when the rule is added to the rule set</td>
</tr>
<tr>
<td>FP</td>
<td>The number of negative documents that are satisfied by the rule set when the rule is added to the rule set</td>
</tr>
<tr>
<td>Support</td>
<td>The number of documents that are satisfied by the rule set when the rule is added to the rule set</td>
</tr>
<tr>
<td>rTP</td>
<td>The number of positive documents that are satisfied by the rule when the rule is added to the rule set</td>
</tr>
<tr>
<td>rFP</td>
<td>The number of negative documents that are satisfied by the rule when the rule is added to the rule set</td>
</tr>
<tr>
<td>rSupport</td>
<td>The number of documents that are satisfied by the rule when the rule is added to the rule set</td>
</tr>
<tr>
<td>F1</td>
<td>The F1 score of the rule set when the rule is added to the rule set</td>
</tr>
<tr>
<td>Precision</td>
<td>The precision of the rule set when the rule is added to the rule set</td>
</tr>
<tr>
<td>Recall</td>
<td>The recall of the rule set when the rule is added to the rule set</td>
</tr>
</tbody>
</table>

This data table contains the discovered rule sets for predicting the target levels of the target variable. In each rule set, the order of the rules is important and helps you interpret the results. The first rule is trained using all the data; the second rule is trained on the data that did not satisfy the first rule; and subsequent rules are built only after the removal of observations that satisfy previous rules. The fit statistics (TP, FP, Support, F1,
Precision, and Recall) of each rule are cumulative and represent totals that include using that particular rule along with all the previous rules in the rule set.

When you specify TARGETTYPE=MULTICLASS in the DOCINFO statement, each target level of the target variable defines a category and the target field contains the same content as the Target_val field. When TARGETTYPE=BINARY in the DOCINFO statement, each target variable defines a category and the target field contains the same content as the Target_var field.

### RULETERMS= Data Table

The RULETERMS= option in the OUTPUT statement specifies a data table to contain the terms in the rules. The information in this data table is used in the scoring phase for scoring documents.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>The target category that the term is selected to model</td>
</tr>
<tr>
<td>Target_var</td>
<td>The variable that contains the target</td>
</tr>
<tr>
<td>Target_val</td>
<td>The value of the target variable</td>
</tr>
<tr>
<td>Ruleid</td>
<td>The ID of a rule (Ruleid starts from 1)</td>
</tr>
<tr>
<td>Ruleid_loc</td>
<td>The ID of a rule in a rule set (in each rule set, Ruleid_loc starts from 1)</td>
</tr>
<tr>
<td>Rule</td>
<td>The text content of the rule</td>
</tr>
<tr>
<td><em>termnum</em></td>
<td>The ID of a term that is used in the rule</td>
</tr>
<tr>
<td>Direction</td>
<td>Specifies whether the term is positive or negative (if Direction=1, the term is positive; if Direction=–1, the term is negative)</td>
</tr>
<tr>
<td>Weight</td>
<td>The weight of a term</td>
</tr>
</tbody>
</table>

Term weights are used for scoring documents. The weight of a negative term is always –1. If a positive term is in rule r and there are k positive terms in the rule, the weight of this positive term is $1/k + 0.000001$. If a document contains all the positive terms in the rule but none of the negative terms, the score of the document is $k \times (1/k + 0.000001) > 1$, indicating that the document satisfies the rule. Otherwise, the document’s score is less than 1, indicating that the document does not satisfy the rule.

### Scoring Data Set

This section describes the output data set that PROC BOOLRULE produces when you specify the corresponding option in the SCORE statement.

### OUTMATCH= Data Table

The OUTMATCH= option in the SCORE statement specifies the output data table to contain the rule-matching results (that is, whether a document satisfies a rule). A document satisfies a rule (in other words, a rule is matched in the document) if and only if all the positive terms in the rule are present in the document and none of the negative terms are present in the document. PROC BOOLRULE also outputs a special rule for which ID=0. If a document satisfies the rule for which ID=0, then the document does not satisfy any rule in the RULETERMS= table. For this special rule, the target has a missing value.
Table 2.5 shows the fields in this data table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Document</em></td>
<td>ID of the document that satisfies the rule</td>
</tr>
<tr>
<td><em>Target</em></td>
<td>ID of the target that the rule is generated for</td>
</tr>
<tr>
<td><em>Rule_ID</em></td>
<td>ID of the rule that the document satisfies</td>
</tr>
</tbody>
</table>

### Examples: BOOLRULE Procedure

#### Example 2.1: Rule Extraction for Binary Targets

This example generates rules for a data table that contains various types of customer reviews. The following DATA step creates the mycas.reviews data table, which contains nine observations that have four variables. The text variable contains the input reviews. The positive variable contains the sentiment of the reviews: a value of 1 indicates that the review is positive, and a value of 0 indicates that the review is negative. The category variable contains the category of the reviews. The did variable contains the ID of the documents. Each row in the data table represents a document for analysis.

```plaintext
data mycas.reviews;
  infile datalines delimiter='|' missover;
  length text $300 category $20;
  input text$ positive category$ did;
  datalines;
  This is the greatest phone ever! love it!|1|electronics|1
  The phone's battery life is too short and screen resolution is low.|0|electronics|2
  The screen resolution is low, but I love this tv.|1|electronics|3
  The movie itself is great and I like it, although the resolution is low.|1|movies|4
  The movie's story is boring and the acting is poor.|0|movies|5
  I watched this movie on tv, it's not good on a small screen. |0|movies|6
  watched the movie first and loved it, the book is even better!|1|books |7
  I like the story in this book, they should put it on screen.|1|books|8
  I love the author, but this book is a waste of time, don't buy it.|0|books|9
; run;
```

The following TEXTMINE procedure call parses the mycas.reviews data table, stores the term-by-document matrix in the mycas.reviews_bow data table in transactional format, and stores terms that appeared in the mycas.reviews data table in the mycas.reviews_terms data table:

```plaintext
proc textmine data=mycas.reviews;
  doc_id
did;
  var
```
The following statements run PROC BOOLRULE to extract rules from the mycas.reviews_bow data table and run PROC PRINT to show the results. By default, TARGETTYPE=BINARY. One target variable, positive, is specified; this variable indicates whether the reviews are positive or negative.

```
proc boolrule
  data = mycas.reviews_bow
  docid = _document_
  termid = _termnum_
  docinfo = mycas.reviews
  terminfo = mycas.reviews_terms
  minsupports = 1
  mpos = 1
  gpos = 1;
  docinfo
    id = did
    targets = (positive);
  terminfo
    id = key
    label = term;
  output
    ruleterms = mycas.ruleterms
    rules = mycas.rules;
run;
```

Output 2.1.1 shows that the mycas.rules data table contains rules that are generated for the “positive” categories.

```
Output 2.1.1  The mycas.rules Data Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>TARGET</th>
<th>RULEID</th>
<th>RULE</th>
<th>F1</th>
<th>PRECISION</th>
<th>RECALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>positive</td>
<td>1</td>
<td>the &amp; ~'s &amp; ~not</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```
Example 2.2: Rule Extraction for a Multiclass Target

This example uses the same input table and the same TEXTMINE procedure call that are used in Example 2.1 to illustrate how you can extract rules for a multiclass target. The DATA step and procedure call are repeated here for convenience.

The following DATA step creates the mycas.reviews data table, which contains nine observations that have four variables. The text variable contains the input reviews. The positive variable contains the sentiment of the reviews: a value of 1 indicates that the review is positive, and a value of 0 indicates that the review is negative. The category variable contains the category of the reviews. The did variable contains the ID of the documents. Each row in the data table represents a document for analysis.

```sas
data mycas.reviews;
  infile datalines delimiter='|' missover;
  length text $300 category $20;
  input text$ positive category$ did;
  datalines;
  This is the greatest phone ever! love it!|1|electronics|1
  The phone's battery life is too short and screen resolution is low.|0|electronics|2
  The movie is great and I like it, although the resolution is low.|1|movies|4
  The movie's story is boring and the acting is poor.|0|movies|5
  I watched this movie on tv, it's not good on a small screen.|0|movies|6
  I like the story in this book, they should put it on screen.|1|books|7
  I love the author, but this book is a waste of time, don't buy it.|0|books|9
run;
```

The following TEXTMINE procedure call parses the mycas.reviews data table, stores the term-by-document matrix in the mycas.reviews_bow data table in transactional format, and stores terms that appeared in the mycas.reviews data table in the mycas.reviews_terms data table:

```sas
proc textmine data=mycas.reviews;
  doc_id did;
  var text;
  parse nonoungroups notagging
    entities = none
    outparent = mycas.reviews_bow
    outterms = mycas.reviews_terms
    reducef = 1;
run;
```

The following statements run PROC BOOLRULE to extract rules from the mycas.reviews_bow data table and run PROC PRINT to show the results. TARGETTYPE=MULTICLASS is specified, and category is specified as the target variable, which contains three levels: "electronics," "movies," and "books." Each level defines a category for which the BOOLRULE procedure extracts rules.
proc boolrule
  data = mycas.reviews_bow
docid = _document_
termid = _termnum_
docinfo = mycas.reviews
termino = mycas.reviews_terms
minsupports = 1
mpos = 1
gpos = 1;
docinfo
  id = did
targettype = multiclass	
targets = (category);
termino
  id = key
tab = term;
output
  ruleterms = mycas.ruleterms
rules = mycas.rules;
run;

data rules;
set mycas.rules;
proc print data=rules;
  var target ruleid rule F1 precision recall;
run;

Output 2.2.1 shows that the mycas.rules data table contains rules that are generated for the “electronics,” “movies,” and “books” categories.

<table>
<thead>
<tr>
<th>Obs</th>
<th>TARGET</th>
<th>RULEID</th>
<th>RULE</th>
<th>F1</th>
<th>PRECISION</th>
<th>RECALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>electronics</td>
<td>3</td>
<td>phone</td>
<td>0.8</td>
<td>1</td>
<td>0.66667</td>
</tr>
<tr>
<td>2</td>
<td>electronics</td>
<td>4</td>
<td>resolution &amp; ~it</td>
<td>1.0</td>
<td>1</td>
<td>1.00000</td>
</tr>
<tr>
<td>3</td>
<td>books</td>
<td>2</td>
<td>book</td>
<td>1.0</td>
<td>1</td>
<td>1.00000</td>
</tr>
<tr>
<td>4</td>
<td>movies</td>
<td>1</td>
<td>movie &amp; ~love</td>
<td>1.0</td>
<td>1</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

Example 2.3: Using Events in Rule Extraction

This example uses the same input table and the same TEXTMINE procedure call that are used in Example 2.1 to illustrate how you can use events in rule extraction. The DATA step and procedure call are repeated here for convenience.

When TARGETTYPE=MULTICLASS, each level of the target variable defines a category for rule extraction. If you want to extract rules for only a subset of the levels of the target variable, you can use the EVENTS= option to specify the categories for which you want to extract rules.

The following DATA step creates the mycas.reviews data table, which contains nine observations that have four variables. The text variable contains the input reviews. The positive variable contains the sentiment of
Example 2.3: Using Events in Rule Extraction

the reviews: a value of 1 indicates that the review is positive and a value of 0 indicates that the review is negative. The category variable contains the category of the reviews. The did variable contains the ID of the documents. Each row in the data table represents a document for analysis.

data mycas.reviews;
  infile datalines delimiter='|' missover;
  length text $300 category $20;
  input text$ positive category$ did;
  datalines;
  This is the greatest phone ever! love it!|1|electronics|1
  The phone's battery life is too short and screen resolution is low.|0|electronics|2
  The screen resolution is low, but I love this tv.|1|electronics|3
  The movie itself is great and I like it, although the resolution is low.|1|movies|4
  The movie's story is boring and the acting is poor.|0|movies|5
  I watched this movie on tv, it's not good on a small screen.|0|movies|6
  watched the movie first and loved it, the book is even better!|1|books|7
  I like the story in this book, they should put it on screen.|1|books|8
  I love the author, but this book is a waste of time, don't buy it.|0|books|9
run;

The following TEXTMINE procedure call parses the mycas.reviews data table, stores the term-by-document matrix in the mycas.reviews_bow data table in transactional format, and stores terms that appeared in the mycas.reviews data table in the mycas.reviews_terms data table:

proc textmine data=mycas.reviews;
  doc_id = did;
  var = text;
  parse
    nonoungroups
    notagging
    entities = none
    outparent = mycas.reviews_bow
    outterms = mycas.reviews_terms
    reducef = 1;
run;

The following statements run PROC BOOLRULE to extract rules from the mycas.reviews_bow data table and run PROC PRINT to show the results. TARGETTYPE=BINARY is specified, and category is specified as the target variable, which contains three levels: “electronics,” “movies,” and “books.” Because the “movies” and “books” levels are specified in the EVENTS= option, PROC BOOLRULE procedure extracts rules for “movies” and “books,” but not “electronics.”

proc boolrule
  data = mycas.reviews_bow
  docid = _document_
  termid = _termnum_
  docinfo = mycas.reviews
  terminfo = mycas.reviews_terms
  minsupports = 1
Chapter 2: The BOOLRULE Procedure

mpos = 1
gpos = 1;
docinfo
  id = did
  targettype = binary
  targets = (category)
  events = ("movies" "books");
terminfo
  id = key
  label = term;
output
  ruleterms = mycas.ruleterms
  rules = mycas.rules;
run;
data rules;
  set mycas.rules;
  proc print data=rules;
    var target ruleid rule F1 precision recall;
  run;

Output 2.3.1 shows that the mycas.rules data table contains rules that are generated for the “movies” and “books” categories.

Output 2.3.1  The mycas.rules Data Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>TARGET</th>
<th>RULEID</th>
<th>RULE</th>
<th>F1</th>
<th>PRECISION</th>
<th>RECALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>category</td>
<td>1</td>
<td>movie</td>
<td>0.8</td>
<td>1</td>
<td>0.66667</td>
</tr>
<tr>
<td>2</td>
<td>category</td>
<td>2</td>
<td>book</td>
<td>1.0</td>
<td>1</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

Example 2.4: Scoring

This example uses the same input table and the same TEXTMINE procedure call that are used in Example 2.1 to illustrate how you can match extracted rules in documents. Then it adds the DATA step to generate testing data. The DATA step and procedure call are repeated here for convenience.

The following DATA step creates the mycas.reviews data table, which contains nine observations that have four variables. The text variable contains the input reviews. The positive variable contains the sentiment of the reviews: a value of 1 indicates that the review is positive, and a value of 0 indicates that the review is negative. The category variable contains the category of the reviews. The did variable contains the ID of the documents. Each row in the data table represents a document for analysis.

data mycas.reviews;
  infile datalines delimiter='|' missover;
  length text $300 category $20;
  input text$ positive category$ did;
  datalines;
  This is the greatest phone ever! love it! |1|electronics|1
  The phone’s battery life is too short and screen resolution is low. |0|electronics|2
  The screen resolution is low, but I love this tv. |1|electronics|3
  The movie itself is great and I like it, although the resolution is low. |1|movies|4
run;
The movie's story is boring and the acting is poor. | 0 | movies | 5
I watched this movie on tv, it's not good on a small screen. | 0 | movies | 6
watched the movie first and loved it, the book is even better! | 1 | books | 7
I like the story in this book, they should put it on screen. | 1 | books | 8
I love the author, but this book is a waste of time, don't buy it. | 0 | books | 9

The following DATA step generates the testing data, which contain two observations that have two variables. The text variable contains the input reviews. The did variable contains the ID of the documents. Each row in the data table represents a document for analysis.

data mycas.reviews_test;
  infile datalines delimiter='|' missover;
  length text $300;
  input text$ did;
  datalines;
  love it! a great phone, even better than advertised| 1
  I like the book, GREATEST in this genre| 2
  run;

The following TEXTMINE procedure call parses the mycas.reviews data table, stores the term-by-document matrix in the mycas.reviews_bow data table in transactional format, and stores terms that appeared in the mycas.reviews data table in the mycas.reviews_terms data table:

proc textmine data=mycas.reviews;
  doc_id = did;
  var = text;
  parse
    nonoungroups
    notagging
    entities = none
    outparent = mycas.reviews_bow
    outterms = mycas.reviews_terms
    outconfig = mycas.parseconfig
    reducef = 1;
  run;

The following statements run PROC BOOLRULE to extract rules from the mycas.reviews_bow data table. TARGETTYPE=BINARY is specified. One target variable, positive, is specified; this variable indicates whether the reviews are positive or negative.

proc boolrule
  data = mycas.reviews_bow
  docid = _document_
  termid = _termnum_
  docinfo = mycas.reviews
  terminfo = mycas.reviews_terms
  minsupports = 1
Chapter 2: The BOOLRULE Procedure

```plaintext
mpos = 1
gpos = 1;
docinfo
  id = did
targettype = binary
targets = (positive);
terminfo
  id = key
  label = term;
output
  ruleterms = mycas.ruleterms
  rules = mycas.rules;
run;
```

The TMSCORE procedure uses the parsing configuration that is stored in the mycas.parseconfig data table to parse the mycas.reviews_test data table. The term-by-document matrix is stored in the mycas.reviews_test_bow data table.

```plaintext
proc tmscore
data = mycas.reviews_test
terms = mycas.reviews_terms
config = mycas.parseconfig
outparent = mycas.reviews_test_bow;
doc_id did;
var text;
run;
```

The following statements run PROC BOOLRULE to match rules in the testing data and run PROC PRINT to show the matching results:

```plaintext
proc boolrule
data = mycas.reviews_test_bow
docid = _document_
termid = _termnum_
score
  ruleterms = mycas.ruleterms
  outmatch = mycas.match;
run;
```

The mycas.match data table in Output 2.4.1 shows which documents satisfy which rules.

```
Output 2.4.1  The mycas.match Data Table

    Obs _DOCUMENT_ _TARGET_ _RULE_ID_
 1       1       0
 2       2       1       1
```

The mycas.match data table in Output 2.4.1 shows which documents satisfy which rules.
Chapter 3
The TEXTMINE Procedure

Contents

Overview: TEXTMINE Procedure ......................................................... 38
PROC TEXTMINE Features ............................................................... 38
Using CAS Sessions and CAS Engine Librefs ...................................... 39
Getting Started: TEXTMINE Procedure ............................................. 40
Syntax: TEXTMINE Procedure ......................................................... 44
PROC TEXTMINE Statement ............................................................ 44
DOC_ID Statement .......................................................................... 45
PARSE Statement ............................................................................ 46
SAVESTATE Statement ..................................................................... 50
SELECT Statement ........................................................................... 50
SVD Statement ................................................................................ 52
TARGET Statement ........................................................................... 56
VARIABLES Statement ...................................................................... 56
Details: TEXTMINE Procedure .......................................................... 56
Natural Language Processing ............................................................. 56
Stemming ......................................................................................... 56
Part-of-Speech Tagging ..................................................................... 56
Noun Group Extraction ..................................................................... 57
Entity Identification ......................................................................... 57
Multiword Terms Handling ............................................................... 59
Language Support ............................................................................ 59
Term and Cell Weighting ................................................................. 60
Sparse Format .................................................................................. 60
Coordinate List (COO) Format .......................................................... 61
Singular Value Decomposition .......................................................... 61
Applications in Text Mining .............................................................. 61
Computation .................................................................................... 62
SVD-Only Mode ............................................................................... 62
Topic Discovery ................................................................................ 62
Output Data Tables .......................................................................... 63
The OUTCHILD= Data Table .......................................................... 63
The OUTCONFIG= Data Table .......................................................... 63
The OUTDOCPRO= Data Table .......................................................... 64
The OUTPARENT= Data Table ............................................................ 64
The OUTPOS= Data Table ................................................................. 65
The OUTTERMS= Data Table ............................................................ 65
Overview: TEXTMINE Procedure

The TEXTMINE procedure integrates natural language processing and statistical analysis to analyze large-scale textual data in SAS Viya. PROC TEXTMINE supports a wide range of fundamental text analysis features, which include tokenizing, stemming, part-of-speech tagging, noun group extraction, default or customized stop lists and start lists, entity parsing, multiword tokens, synonym lists, term weighting, term-by-document matrix creation, dimension reduction with singular value decomposition (SVD), and topic discovery. The procedure leverages the tmMine action of the textMining action set to accomplish these tasks, but it does not surface all of the action’s capabilities. Further functionality is available to you if you call this action directly using PROC CASL.

PROC TEXTMINE Features

The TEXTMINE procedure processes large-scale textual data in parallel in order to achieve efficiency and scalability. The following list summarizes the basic features of PROC TEXTMINE:

- Functionalities that are related to document parsing, term-by-document matrix creation, and dimension reduction are integrated into one procedure in order to process data more efficiently.

- Parsing supports essential natural language processing (NLP) features, which include tokenizing, stemming, part-of-speech tagging, noun group extraction, default or customized stop lists and start lists, entity parsing, multiword tokens, and synonym lists.

- Term weighting and filtering are supported for term-by-document matrix creation.

- Parsing and term-by-document matrix creation are processed in parallel.

- Computation of singular value decomposition (SVD) is parallelized.

- Topic discovery is integrated into the procedure.

- All phases of processing use a high degree of multithreading.
Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the `mysess` session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 1 in Chapter 1, “Shared Concepts.”
Chapter 3: The TEXTMINE Procedure

Getting Started: TEXTMINE Procedure

The input data must be a table on your CAS server, and a CAS session must be set up. For more information, see the sections “Using CAS Sessions and CAS Engine Librefs” on page 1 and “Loading a SAS Data Set onto a CAS Server” on page 2 in Chapter 1, “Shared Concepts.”

The following DATA step creates the getstart data table, which contains 16 observations that have two variables, in your CAS session. The text variable contains the input documents, and the did variable contains the ID of the documents. Each row in the data table represents a document for analysis.

```sas
data mycas.getstart;
  infile datalines delimiter='|' missover;
  length text $150;
  input text$ did;
  datalines;
  Reduces the cost of maintenance. Improves revenue forecast. | 1
  Analytics holds the key to unlocking big data. | 2
  The cost of updates between different environments is eliminated. | 3
  Ensures easy deployment in the cloud or on-site. | 4
  Organizations are turning to SAS for business analytics. | 5
  This removes concerns about maintenance and hidden costs. | 6
  Service-oriented and cloud-ready for many cloud infrastructures. | 7
  Easily apply machine learning and data mining techniques to data. | 8
  SAS Viya will address data analysis, modeling and learning. | 9
  Helps customers reduce cost and make better decisions faster. | 10
  Simple, powerful architecture ensures easy deployment in the cloud. | 11
  SAS is helping industries glean insights from data. | 12
  Solve complex business problems faster than ever. | 13
  Shatter the barriers associated with data volume with SAS Viya. | 14
  Casual business users, data scientists and application developers. | 15
  Serves as the basis for innovation causing revenue growth. | 16
run;
```

These statements assume that your CAS engine libref is named mycas, but you can substitute any appropriately defined CAS engine libref.

The following DATA step uses the default stop list to eliminate noisy, noninformative terms:

```sas
proc cas;
  loadtable caslib="ReferenceData" path="en_stoplist.sashdat";
run;
quit;
```

The following statements parse the input collection and use singular value decomposition followed by a rotation to discover topics that exist in the sample collection. The statements specify that all terms in the document collection, except for those on the stop list, are to be kept for generating the term-by-document matrix. The summary information about the terms in the document collection is stored in a data table named mycas.terms. The SVD statement requests that the first three singular values and singular vectors be computed. The topic assignments of the documents are stored in a data table named mycas.docpro, and the descriptive terms that define each topic are stored in a data table named mycas.topics.
proc textmine data=mycas.getstart;
   doc_id did;
   variables text;
   parse outterms = mycas.terms
               reducef = 1
               stop = mycas.en_stoplist;
   svd k = 3
        outdocpro = mycas.docpro
        outtopics = mycas.topics;
   savestate rstore = mycas.aStoreTab;
run;

The output from this analysis is presented in Figure 3.2, Figure 3.3 and Figure 3.4.

Figure 3.1 shows the SAS log that is generated by PROC TEXTMINE; the log provides information about the default configurations used by the procedure and about the input and output files including the number of observations in each of the output tables. The mycas.terms data table lists the discovered terms. The mycas.docpro data table contains four variables: the first variable is the document ID, and the remaining three variables are obtained by projecting the original document onto the three left-singular vectors that have been rotated with the default orthogonal (varimax) rotation. The mycas.topics data table has 3 variables containing summary information of the discovered topics. Finally, the mycas.astoretab table contains a binary representation of a scoring model.

NOTE: Stemming will be used in parsing.
NOTE: Tagging will be used in parsing.
NOTE: Noun groups will be used in parsing.
NOTE: No TERMWGT option is specified. TERMWGT=ENTROPY will be run by default.
NOTE: No CELLWGT option is specified. CELLWGT=LOG will be run by default.
NOTE: No ENTITIES option is specified. ENTITIES=NONE will be run by default.
NOTE: Topics have been requested so the document unit normalization will not occur unless requested.
NOTE: The dense SVD solver was used for this calculation.
NOTE: Wrote 12532 bytes to the savestate file ASTORETAB.
NOTE: The Cloud Analytic Services server processed the request in 1.670414 seconds.
NOTE: The data set MYCAS.TERMS has 134 observations and 11 variables.
NOTE: The data set MYCAS.DOCPRO has 16 observations and 4 variables.
NOTE: The data set MYCAS.TOPICS has 3 observations and 3 variables.
NOTE: The data set MYCAS.ASTORETAB has 1 observations and 2 variables.

The following statements use PROC PRINT in Base SAS to show the contents of the first 10 rows of the sorted mycas.docpro data table that is generated by the TEXTMINE procedure:
data docpro;
  set mycas.docpro;
run;
proc sort data=docpro;
  by did;
run;
proc print data = docpro (obs=10);
run;

Figure 3.2 shows the output of PROC PRINT. For information about the output of the OUTDOCPRO= option, see the section “The OUTDOCPRO= Data Table” on page 64.

Figure 3.2 The mycas.docpro Data Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>did</th>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.7460570931</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.1111856451</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0.0964494952</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.8688770161</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.4742893251</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0.6276285113</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0.0901933118</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0.0626896657</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>0.5236329356</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0.0478786576</td>
<td>0.0703302315</td>
</tr>
</tbody>
</table>

The following statements use a DATA step and PROC PRINT to show the contents of the mycas.topics data table that is generated by the TEXTMINE procedure:

data topics; set mycas.topics; run;
proc print data = topics;
run;

Figure 3.3 shows the output of PROC PRINT. The three discovered topics are listed with four descriptive terms to characterize each topic.

Figure 3.3 The mycas.topics Data Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>_topicid</th>
<th>_name</th>
<th>_termCutOff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>easy deployment, deployment, +ensure, easy, cloud</td>
<td>0.135</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>sas, data, viya, analytics, +industry</td>
<td>0.149</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>+cost, maintenance, revenue forecast, forecast, +improve</td>
<td>0.146</td>
</tr>
</tbody>
</table>

The following statements use a DATA step and the SORT and PRINT procedures to show the first 10 observations of the mycas.terms data table that is generated by the TEXTMINE procedure:

data terms; set mycas.terms; run;
proc sort data = terms; by key; run;
proc print data = terms (obs=10);
  var term role freq numdocs key parent;
run;
Figure 3.4 shows the output of PROC PRINT, which provides details about the terms that are identified by the TEXTMINE procedure. Only the values of the variables term, role, freq, numdocs, key, and parent are displayed. For information about the output of the OUTTERMS= option, see the section “The OUTTERMS= Data Table” on page 65.

![Figure 3.4 The mycas.terms Data Table](image)

The following DATA step and statements create data and then score that data with PROC ASTORE.

```plaintext
data mycas.scoreData;
  infile datalines delimiter='|' missover;
  length text $150;
  input text$ id;
  datalines;
  Deployment in the cloud or on-site. | 1
  SAS for business analytics. | 2
  Maintenance and hidden costs. | 3
run;

proc astore;
  score rstore=mycas.aStoreTab
    data=mycas.scoreData
    out= mycas.scoreResults
    copyVars= id;
run;

proc sort data=mycas.scoreResults out=scoreResults;
  by id;
run;
proc print data = scoreResults;
run;
```

Figure 3.5 shows the output of PROC PRINT, which provides the topic score for the documents processed by the ASTORE PROCEDURE.
Syntax: TEXTMINE Procedure

The following statements are available in the TEXTMINE procedure:

```r
PROC TEXTMINE DATA=CAS-libref.data-table <options> ;
  VARIABLES variable ;
  TARGET variable ;
  DOC_ID variable ;
  PARSE <parse-options> ;
  SELECT label-list <GROUP=group-option> KEEP | IGNORE ;
  SVD <svd-options> ;
  SAVESTATE RSTORE=CAS-libref.data-model ;
```

The PROC TEXTMINE statement, the VARIABLES statement, and the DOC_ID statement are required.

The following sections describe the PROC TEXTMINE statement and then describe the other statements in alphabetical order.

PROC TEXTMINE Statement

```r
PROC TEXTMINE DATA=CAS-libref.data-table <options> ;
```

The PROC TEXTMINE statement invokes the procedure. Table 3.1 summarizes the options in the statement by function. The options are then described fully in alphabetical order.

<table>
<thead>
<tr>
<th>option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Basic Options</strong></td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>DOC=</td>
</tr>
<tr>
<td>LANGUAGE=</td>
<td>Specifies the language that the input data table of documents uses</td>
</tr>
<tr>
<td>NEWVARNAMES</td>
<td>Specifies that the new-style variable names should be used on tables</td>
</tr>
<tr>
<td><strong>Multithreading Options</strong></td>
<td></td>
</tr>
<tr>
<td>NTHREADS=</td>
<td>Specifies number of threads</td>
</tr>
</tbody>
</table>
You must specify the following option:

**DATA=** `CAS-libref data-table`

names the input data table for PROC TEXTMINE to use. The default is the most recently created data table. *CAS-libref.data-table* is a two-level name, where

*CAS-libref* refers to a collection of information that is defined in the LIBNAME statement and includes the `caslib`, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about *CAS-libref*, see the section “Using CAS Sessions and CAS Engine Librefs” on page 39.

*data-table* specifies the name of the input data table.

Each row of the input data table must contain one text variable and one ID variable that correspond to the text and the unique ID of a document, respectively.

When you specify the SVD statement but not the PARSE statement, PROC TEXTMINE runs in SVD-only mode. In this mode, the DATA= option names the input SAS data table that contains the term-by-document matrix that is generated by the OUTPARENT= option in the PARSE statement.

You can also specify the following *options*:

**LANGUAGE=** `language`

names the language that is used by the documents in the input SAS data table. Languages supported in the current release are Arabic, Chinese, Croatian, Czech, Danish, Dutch, English, Farsi, Finnish, French, German, Greek, Hebrew, Hungarian, Indonesian, Italian, Japanese, Kazakh, Korean, Norwegian, Polish, Portuguese, Russian, Slovak, Slovene, Spanish, Swedish, Thai, Turkish and Vietnamese. By default, LANGUAGE=ENGLISH.

**NEWVARNAMES**

adds leading and trailing blanks to variable names in the input and output tables.

**NTHREADS=** `nthreads`

specifies the number of threads to be used. By default, the number of threads is the same as the number of CPUs on the CAS server.

---

**DOC_ID Statement**

```
DOC_ID variable ;
```

The DOC_ID statement specifies the *variable* that contains the ID of each document. In the input data table, each row corresponds to one document. The ID of each document must be unique; it can be either a number or a string of characters.
Chapter 3: The TEXTMINE Procedure

**PARSE Statement**

```
PARSE <parse-options> ;
```

The PARSE statement specifies the options for parsing the input documents and creating the term-by-document matrix. Table 3.2 summarizes the `parse-options` in the statement by function. The `parse-options` are then described fully in alphabetical order.

<table>
<thead>
<tr>
<th><code>parse-option</code></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parsing Options</strong></td>
<td></td>
</tr>
<tr>
<td>ENTITIES=</td>
<td>Specifies whether to extract entities in parsing</td>
</tr>
<tr>
<td>MULTITERM=</td>
<td>Specifies the multiword term list</td>
</tr>
<tr>
<td>NONOUNGROUPS</td>
<td>Suppresses noun group extraction in parsing</td>
</tr>
<tr>
<td>NONG</td>
<td>Suppresses stemmin in parsing</td>
</tr>
<tr>
<td>NOTAGGING</td>
<td>Suppresses part-of-speech tagging in parsing</td>
</tr>
<tr>
<td>SHOWDROPPEDTERMS=</td>
<td>Includes dropped terms in the <code>OUTTERMS=</code> data table</td>
</tr>
<tr>
<td>START=</td>
<td>Specifies the start list</td>
</tr>
<tr>
<td>STOP=</td>
<td>Specifies the stop list</td>
</tr>
<tr>
<td>SYNONYM</td>
<td>Specifies the synonym list</td>
</tr>
<tr>
<td><strong>Term-by-Document Matrix Creation Options</strong></td>
<td></td>
</tr>
<tr>
<td>CELLWGT=</td>
<td>Specifies how cells are weighted</td>
</tr>
<tr>
<td>REDUCEF=</td>
<td>Specifies the frequency for term filtering</td>
</tr>
<tr>
<td>TERMWGT=</td>
<td>Specifies how terms are weighted</td>
</tr>
<tr>
<td><strong>Output Options</strong></td>
<td></td>
</tr>
<tr>
<td>OUTCHILD=</td>
<td>Specifies the data table to contain the raw term-by-document matrix. All kept terms, whether or not they are child terms, are represented in this data table along with their corresponding frequency.</td>
</tr>
<tr>
<td>OUTCONFIG=</td>
<td>Specifies the data table to contain the option settings that PROC TEXTMINE uses in the current run</td>
</tr>
<tr>
<td>OUTPARENT=</td>
<td>Specifies the data table to contain the term-by-document matrix. Child terms are not represented in this data table. The frequencies of child terms are attributed to their corresponding parents.</td>
</tr>
<tr>
<td>OUTTERMS=</td>
<td>Specifies the data table to contain the summary information about the terms in the document collection</td>
</tr>
<tr>
<td>OUTPOS=</td>
<td>Specifies the data table to contain the position information about the child terms’ occurrences in the document collection</td>
</tr>
</tbody>
</table>

You can specify the following `parse-options`. 
CELLWGT=LOG | NONE
specifies how the elements in the term-by-document matrix are weighted. You can specify the following values:

- **LOG** weights cells by using the log formulation. For information about the log formulation for cell weighting, see the section “Term and Cell Weighting” on page 60.
- **NONE** specifies that no cell weight be applied.

ENTITIES=STD | NONE
determines whether to use the standard LITI file for entity extraction. You can specify the following values:

- **STD** uses the standard LITI file for entity extraction. A term such as “George W. Bush” is recognized as an entity and given the corresponding entity role and attribute. For this term, the entity role is PERSON and the attribute is Entity. Although the entity is treated as the single term, “george w. bush,” the individual tokens “george,” “w,” and “bush” are also included.
- **NONE** does not use the standard LITI file for entity extraction.

By default, ENTITIES=NONE.

MULTITERM=CAS-libref.data-table
specifies the input SAS data table that contains a list of multiword terms. **CAS-libref.data-table** is a two-level name, where **CAS-libref** refers to the caslib and session identifier, and **data-table** specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39. The multiword terms are case-sensitive and are treated as a single entry by the TEXTMINE procedure. Thus, the terms “Thank You” and “thank you” are processed differently. Consequently, you must convert all text strings to lowercase or add each of the multiterm’s case variations to the list before using the TEXTMINE procedure to create consistent multiword terms. The multiterm data table must have a variable **Multiterm** and each of its values must be formatted in the following manner:

```
multiterm: 3: pos
```

Specifically, the first item is the multiword term itself followed by a colon, the second item is a number that represents the token type followed by a colon, and the third item is the part of speech that the multiword term represents. **NOTE:** The token type 3 is the most common token type for multiterm lists; it represents compound words.

NONOUNGROUPS
NONG
suppresses standard noun group extraction. By default, the TEXTMINE procedure extracts noun groups, returns noun phrases without determiners or prepositions, and (unless the NOSTEMMING option is specified) stems noun group elements.
NOSTEMMING
suppresses stemming of words. By default, words are stemmed; that is, terms such as “advises” and “advising” are mapped to the parent term “advise.” The TEXTMINE procedure uses dictionary-based stemming (also known as lemmatization).

NOTAGGING
suppresses tagging of terms. By default, terms are tagged and the TEXTMINE procedure identifies a term’s part of speech based on context clues. The identified part of speech is provided in the Role variable of the OUTTERMS= data table.

OUTCHILD= **CAS-libref.data-table**
specifies the output data table to contain a compressed representation of the sparse term-by-document matrix. **CAS-libref.data-table** is a two-level name, where **CAS-libref** refers to the caslib and session identifier, and **data-table** specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39. The term counts are not weighted. The data table saves only the kept, representative terms. The child frequencies are not attributed to their corresponding parent (as they are in the OUTPARENT= data table). For more information about the compressed representation of the sparse term-by-document matrix, see the section “The OUTCHILD= Data Table” on page 63.

OUTCONFIG= **CAS-libref.data-table**
specifies the output data table to contain configuration information that is used for the current run of PROC TEXTMINE. **CAS-libref.data-table** is a two-level name, where **CAS-libref** refers to the caslib and session identifier, and **data-table** specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39. The primary purpose of this data table is to relay the configuration information from the TEXTMINE procedure to the TMSCORE procedure. The TMSCORE procedure uses options that are consistent with the TEXTMINE procedure. Thus, the data table that is created by using the OUTCONFIG= option becomes an input data table for PROC TMSCORE and ensures that the parsing options are consistent between the two runs. For more information about this data table, see the section “The OUTCONFIG= Data Table” on page 63.

OUTPARENT= **CAS-libref.data-table**
specifies the output data table to contain a compressed representation of the sparse term-by-document matrix. **CAS-libref.data-table** is a two-level name, where **CAS-libref** refers to the caslib and session identifier, and **data-table** specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39. The term counts can be weighted, if requested. The data table contains only the kept, representative terms, and the child frequencies are attributed to the corresponding parent. To obtain information about the children, use the OUTCHILD= option. For more information about the compressed representation of the sparse term-by-document matrix, see the section “The OUTPARENT= Data Table” on page 64.

OUTPOS= **CAS-libref.data-table**
specifies the output data table to contain the position information about the child terms’ occurrences in the document collection. **CAS-libref.data-table** is a two-level name, where **CAS-libref** refers to the caslib and session identifier, and **data-table** specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39. For more information about this data table, see the section “The OUTPOS= Data Table” on page 65.
OUTTERMS= CAS-libref.data-table
specifies the output data table to contain the summary information about the terms in the document collection. The maximum output length of a tokenized term is 256 bytes. So tokens consisting of an extremely long sequence of letters, numbers and symbols will be truncated to less than or equal to that maximum value of 256 bytes. CAS-libref.data-table is a two-level name, where CAS-libref refers to the caslib and session identifier, and data-table specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39. For more information about this data table, see the section “Output Data Tables” on page 63.

REDUCEF=n
removes terms that are not in at least n documents. The value of n must be a positive integer. By default, REDUCEF=4.

SHOWDROPPEDTERMS
includes the terms that have a keep status of N in the OUTTERMS= data table and the OUTCHILD= data table.

START= CAS-libref.data-table
specifies the input data table that contains the terms that are to be kept for the analysis. CAS-libref.data-table is a two-level name, where CAS-libref refers to the caslib and session identifier, and data-table specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39. These terms are displayed in the OUTTERMS= data table with a keep status of Y. All other terms are displayed with a keep status of N if the SHOWDROPPEDTERMS option is specified or not displayed if the SHOWDROPPEDTERMS option is not specified. The START= data table must have a Term variable and can also have a Role variable. You cannot specify both the START= and STOP= options.

STOP= CAS-libref.data-table
specifies the input data table that contains the terms to exclude from the analysis. CAS-libref.data-table is a two-level name, where CAS-libref refers to the caslib and session identifier, and data-table specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39. These terms are displayed in the OUTTERMS= data table with a keep status of N if the SHOWDROPPEDTERMS option is specified. The terms are not identified as parents or children. The STOP= data table must have a Term variable and can also have a Role variable. You cannot specify both the START= and STOP= options.

SYNONYM= CAS-libref.data-table
SYN= CAS-libref.data-table
specifies the input data table that contains user-defined synonyms to be used in the analysis. CAS-libref.data-table is a two-level name, where CAS-libref refers to the caslib and session identifier, and data-table specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39. This data table overrides any relationships that are
identified when terms are stemmed. (Terms are stemmed by default; you can suppress stemming by specifying the NOSTEMMING option.)

**TERMWGT=ENTROPY | MI | NONE**

specifies how terms are weighted. You can specify the following values:

- **ENTROPY** uses the entropy formulation to weight terms.
- **MI** uses the mutual information formulation to weight terms (you must also specify the TARGET statement).
- **NONE** requests that no term weight be applied.

For more information about the entropy formulation and the mutual information formulation for term weighting, see the section “Term and Cell Weighting” on page 60.

**SAVESTATE Statement**

```
SAVESTATE RSTORE=CAS-libref.data-model ;
```

The SAVESTATE statement saves a text mining model to a binary object contained in a data table. The object is referred to as the analytic store and contains the necessary information for scoring a text mining model by the ASTORE procedure. Only complete text models consisting of both parsing and document projections can be saved to the analytic store by the TEXTMINE procedure.

You must specify the following option:

- **RSTORE=** `CAS-libref.data-model`
  
  specifies a data table in which to save the text mining model. `CAS-libref.data-table` is a two-level name, where `CAS-libref` refers to the caslib and session identifier, and `data-table` specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39.

**SELECT Statement**

```
SELECT label-list /< GROUP=group-option> KEEP | IGNORE ;
```

The SELECT statement enables you to specify the parts of speech or entities or attributes that you want to include in or exclude from your analysis. Exclusion by the SELECT statement is different from exclusion that is indicated by the _keep variable in the OUTTERMS= data table. Terms that are excluded by the SELECT statement cannot be included in the OUTTERMS= data table, whereas terms that have _keep=N can be included in the OUTTERMS= data table if the SHOWDROPPEDTERMS option is specified. Terms excluded by the SELECT statement are excluded from the OUTPOS= data table, but terms that have _keep=N are included in OUTPOS= data table. Table 3.3 summarizes the options you can specify in the SELECT statement. The options are then described fully in syntactic order.
Table 3.3  SELECT Statement Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>label-list</td>
<td>Specifies one or more labels of terms that are to be ignored or kept in your analysis</td>
</tr>
<tr>
<td>GROUP=</td>
<td>Specifies whether the labels are parts of speech, entities, or attributes</td>
</tr>
<tr>
<td>IGNORE</td>
<td>Ignores terms whose labels are specified in the label-list</td>
</tr>
<tr>
<td>KEEP</td>
<td>Keeps terms whose labels are specified in the label-list</td>
</tr>
</tbody>
</table>

You must specify a label-list and either the IGNORE or KEEP option:

**label-list**

specifies one or more labels that are either parts of speech or entities or attributes. Each label must be surrounded by double quotation marks and separated by spaces from other labels. Labels are case-insensitive. Terms that have these labels are either ignored during parsing (when the IGNORE option is specified) or kept in the parsing results in the OUTPOS= and OUTTERMS= data tables (when the KEEP option is specified). Table 3.5 shows all possible part-of-speech tags. Table 3.6 shows all valid English entities. The attribute variable in Table 3.11 shows all possible attributes.

**IGNORE**

ignores during parsing all terms whose labels are specified in the label-list, but keeps all other terms in the parsing results (the OUTPOS= and OUTTERMS= data tables).

**KEEP**

keeps in the parsing results (the OUTPOS= and OUTTERMS= data tables) only the terms whose labels are specified in the label-list.

You can also specify the following option:

**GROUP=“ATTRIBUTES” | “ENTITIES” | “POS”**

specifies whether the labels are attributes, entities, or parts of speech. The group type must be surrounded by double quotation marks and is case-insensitive. All labels that are specified in the label-list in the same SELECT statement should belong to the specified group. If you need to select labels from more than one group, you can use multiple SELECT statements (one for each group that you need to select from). You cannot specify multiple SELECT statements for the same group. By default, Num and Punct in the “ATTRIBUTES” group are ignored, but this default is overridden by a SELECT statement that specifies GROUP=“ATTRIBUTES”. By default, GROUP=“POS”.

---

You must specify a label-list and either the IGNORE or KEEP option:

**label-list**

specifies one or more labels that are either parts of speech or entities or attributes. Each label must be surrounded by double quotation marks and separated by spaces from other labels. Labels are case-insensitive. Terms that have these labels are either ignored during parsing (when the IGNORE option is specified) or kept in the parsing results in the OUTPOS= and OUTTERMS= data tables (when the KEEP option is specified). Table 3.5 shows all possible part-of-speech tags. Table 3.6 shows all valid English entities. The attribute variable in Table 3.11 shows all possible attributes.

**IGNORE**

ignores during parsing all terms whose labels are specified in the label-list, but keeps all other terms in the parsing results (the OUTPOS= and OUTTERMS= data tables).

**KEEP**

keeps in the parsing results (the OUTPOS= and OUTTERMS= data tables) only the terms whose labels are specified in the label-list.

You can also specify the following option:

**GROUP=“ATTRIBUTES” | “ENTITIES” | “POS”**

specifies whether the labels are attributes, entities, or parts of speech. The group type must be surrounded by double quotation marks and is case-insensitive. All labels that are specified in the label-list in the same SELECT statement should belong to the specified group. If you need to select labels from more than one group, you can use multiple SELECT statements (one for each group that you need to select from). You cannot specify multiple SELECT statements for the same group. By default, Num and Punct in the “ATTRIBUTES” group are ignored, but this default is overridden by a SELECT statement that specifies GROUP=“ATTRIBUTES”. By default, GROUP=“POS”.
SVD Statement

\texttt{SVD < svd-options> ;}

The SVD statement specifies the options for calculating a truncated singular value decomposition (SVD) of the large, sparse term-by-document matrix that is created during the parsing phase of PROC TEXTMINE. Table 3.4 summarizes the \textit{svd-options} in the statement by function. The \textit{svd-options} are then described fully in alphabetical order.

### Table 3.4 SVD Statement Options

<table>
<thead>
<tr>
<th>svd-option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input Options</strong></td>
<td></td>
</tr>
<tr>
<td>COL=</td>
<td>Specifies the column variable, which contains the column indices of the term-by-document matrix, which is stored in coordinate list (COO) format</td>
</tr>
<tr>
<td>ROW=</td>
<td>Specifies the row variable, which contains the row indices of the term-by-document matrix, which is stored in COO format</td>
</tr>
<tr>
<td>ENTRY=</td>
<td>Specifies the entry variable, which contains the entries of the term-by-document matrix, which is stored in COO format</td>
</tr>
<tr>
<td><strong>SVD Computation Options</strong></td>
<td></td>
</tr>
<tr>
<td>K=</td>
<td>Specifies the number of dimensions to be extracted</td>
</tr>
<tr>
<td>MAX_K=</td>
<td>Specifies the maximum number of dimensions to be extracted</td>
</tr>
<tr>
<td>TOL=</td>
<td>Specifies the maximum allowable tolerance for the singular value</td>
</tr>
<tr>
<td>RESOLUTION</td>
<td>RES=</td>
</tr>
<tr>
<td><strong>Topic Discovery Options</strong></td>
<td></td>
</tr>
<tr>
<td>NUMLABELS=</td>
<td>Specifies the number of terms to be used in the descriptive label for each topic</td>
</tr>
<tr>
<td>ROTATION=</td>
<td>Specifies the type of rotation to be used for topic discovery</td>
</tr>
<tr>
<td>IN_TERMS=</td>
<td>Specifies the data table that contains the terms for topic discovery in SVD-only mode</td>
</tr>
<tr>
<td>EXACTWEIGHT</td>
<td>Prevents rounding of the topic weights</td>
</tr>
<tr>
<td>NOCUTOFFS</td>
<td>Prevents setting term weights to 0 when they are below the threshold</td>
</tr>
<tr>
<td><strong>Output Options</strong></td>
<td></td>
</tr>
<tr>
<td>SVDU=</td>
<td>Specifies the U matrix, which contains the left singular vectors</td>
</tr>
<tr>
<td>SVDV=</td>
<td>Specifies the V matrix, which contains the right singular vectors</td>
</tr>
<tr>
<td>SVDS=</td>
<td>Specifies the S matrix, whose diagonal elements are the singular values</td>
</tr>
<tr>
<td>OUTDOCPRO=</td>
<td>Specifies the data table to contain the projections of the documents</td>
</tr>
<tr>
<td>OUTTOPICS=</td>
<td>Specifies the data table to contain the topics that have been discovered</td>
</tr>
</tbody>
</table>

You can specify the following \textit{svd-options}:
COL=variable
specifies the variable that contains the column indices of the term-by-document matrix. You must specify this option when you run PROC TEXTMINE in SVD-only mode (that is, when you specify the SVD statement but not the PARSE statement).

ENTRY=variable
specifies the variable that contains the entries of the term-by-document matrix. You must specify this option when you run PROC TEXTMINE in SVD-only mode (that is, when you specify the SVD statement but not the PARSE statement).

EXACTWEIGHT
requests that the weights aggregated during topic derivation not be rounded. By default, the calculated weights are rounded to the nearest 0.001.

IN_TERMS=CAS-libref.data-table
specifies the input data table that contains information about the terms in the document collection. CAS-libref.data-table is a two-level name, where CAS-libref refers to the caslib and session identifier, and data-table specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39. The data table should have the variables that are described in Table 3.11. The terms are required to generate topic names in the OUTTOPICS= data table. This option is only for topic discovery in SVD-only mode. This option conflicts with the PARSE statement, and only one of the two can be specified. If you want to run SVD-only mode without topic discovery, then you do not need to specify this option.

K=k
specifies the number of columns in the matrices U, V, and S. This value is the number of dimensions of the data table after SVD is performed. If the value of k is too large, then the TEXTMINE procedure runs for an unnecessarily long time. This option takes precedence over the MAX_K= option. This option also controls the number of topics that are extracted from the text corpus when the ROTATION= option is specified.

MAX_K=n
specifies the maximum value that the TEXTMINE procedure should return as the recommended value of k (the number of columns in the matrices U, V, and S) when the RESOLUTION= option is specified (to recommend the value of k). The TEXTMINE procedure attempts to calculate k dimensions (as opposed to recommending it) when it performs SVD. This option is ignored if the K= option has been specified. This option also controls the number of topics that are extracted from the text corpus when the ROTATION= option is specified.

NOCUTOFFS
uses all weights in the U matrix to form the document projections. When topics are requested, weights below the term cutoff (as calculated in the OUTTOPICS= data table) are set to 0 before the projection is formed.
NUMLABELS=$n$

specifies the number of terms to use in the descriptive label for each topic. The descriptive label provides a quick synopsis of the discovered topics. The labels are stored in the OUTTOPICS= data table. By default, NUMLABELS=5.

OUTDOCPRO=CAS-libref.data-table <KEEPVARIABLES=variable-list><NONORMDOC>

specifies the output data table to contain the projections of the columns of the term-by-document matrix onto the columns of $U$. CAS-libref.data-table is a two-level name, where CAS-libref refers to the caslib and session identifier, and data-table specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39. Because each column of the term-by-document matrix corresponds to a document, the output forms a new representation of the input documents in a space that has much lower dimensionality.

You can copy the variables from the data table that is specified in the DATA= option in the PROC TEXTMINE statement to the data table that is specified in this option. You can specify the following suboptions:

KEEPVARIABLES=variable-list

attaches the content of the variables that are specified in the variable-list to the output. These variables must appear in the data table that is specified in the DATA= option in the PROC TEXTMINE statement.

NONORMDOC

suppresses normalization of the columns that contain the projections of documents to have a unit norm.

OUTTOPICS=CAS-libref.data-table

specifies the output data table to contain the topics that are discovered. CAS-libref.data-table is a two-level name, where CAS-libref refers to the caslib and session identifier, and data-table specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39.

RESOLUTION=LOW | MED | HIGH

RES=LOW | MED | HIGH

specifies how to calculate the recommended number of dimensions (resolution) for the singular value decomposition. If you specify this option, you must also specify the MAX_K= option. A low-resolution singular value decomposition returns fewer dimensions than a high-resolution singular value decomposition. This option recommends the value of $k$ (the number of columns in the matrices $U$, $V$, and $S$) heuristically based on the value specified in the MAX_K= option. Assume that the MAX_K= option is set to $n$ and a singular value decomposition that has $n$ dimensions accounts for $t\%$ of the total variance. You can specify the following values:

HIGH always recommends the maximum number of dimensions; that is, $k = n$.

MED recommends a $k$ that explains $(5/6) \times t\%$ of the total variance.

LOW recommends a $k$ that explains $(2/3) \times t\%$ of the total variance.

By default, RESOLUTION=HIGH.
**ROTATION=VARIMAX | PROMAX**
specifies the type of rotation to be used in order to maximize the explanatory power of each topic. You can specify the following values:

**PROMAX** does an oblique rotation on the original left singular vectors and generates topics that might be correlated.

**VARIMAX** does an orthogonal rotation on the original left singular vectors and generates uncorrelated topics.

By default, ROTATION=VARIMAX.

**ROW=variable** specifies the **variable** that contains the row indices of the term-by-document matrix. You must specify this option when you run PROC TEXTMINE in SVD-only mode (that is, when you specify the SVD statement but not the PARSE statement).

**SVDS=CAS-libref.data-table** specifies the output data table to contain the calculated singular values. **CAS-libref.data-table** is a two-level name, where **CAS-libref** refers to the caslib and session identifier, and **data-table** specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39.

**SVDU=CAS-libref.data-table** specifies the data table to contain the calculated left singular vectors. **CAS-libref.data-table** is a two-level name, where **CAS-libref** refers to the caslib and session identifier, and **data-table** specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39.

**SVDV=CAS-libref.data-table** specifies the data table to contain the calculated right singular vectors. **CAS-libref.data-table** is a two-level name, where **CAS-libref** refers to the caslib and session identifier, and **data-table** specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 39.

**TOL=ε** specifies the maximum allowable tolerance for the singular value. Let $A$ be a matrix. Suppose $\lambda_i$ is the $i$th singular value of $A$ and $\xi_i$ is the corresponding right singular vector. The SVD computation terminates when for all $i \in \{1, \ldots, k\}$, $\lambda_i$ and $\xi_i$ satisfy $\|A^T A \xi_i - \lambda_i \xi_i \|^2 \leq \epsilon$. The default value of $\epsilon$ is $10^{-6}$, which is more than adequate for most text mining problems.
TARGET Statement

TARGET variable ;

This statement specifies the variable that contains the information about the category that a document belongs to. The target variable can be any nominal or ordinal variable; it is used in calculating mutual information term weighting.

VARIABLES Statement

VARIABLES variable ;
VAR variable ;

This statement specifies the variable that contains the text to be processed.

Details: TEXTMINE Procedure

Natural Language Processing

Natural language processing (NLP) techniques can be used to extracting meaningful information from natural language input. The following sections describe features from SAS linguistic technologies that the TEXTMINE procedure implements to support natural language processing.

Stemming

Stemming (a special case of morphological analysis) identifies the possible root form of an inflected word. For example, the word “talk” is the stem of the words “talk,” “talks,” “talking,” and “talked.” In this case “talk” is the parent, and “talk,” “talks,” “talking,” and “talked” are its children. The TEXTMINE procedure uses dictionary-based stemming (also known as lemmatization), which unlike tail-chopping stemmers, produces only valid words as stems. When part-of-speech tagging is on (that is, the NOTAGGING option is not specified), the stem selection process restricts the stem to be of the same part-of-speech as the original term.

Part-of-Speech Tagging

Part-of-speech tagging uses SAS linguistic technologies to identify or disambiguate the grammatical category of a word by analyzing it within its context. For example:

I like to bank at the local branch of my bank.

In this case, the first “bank” is tagged as a verb (V), and the second “bank” is tagged as a noun (N). Table 3.5 shows all possible part-of-speech tags.
Table 3.5 All Part-of-Speech Tags

<table>
<thead>
<tr>
<th>Part-of-Speech Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Adjective</td>
</tr>
<tr>
<td>ADV</td>
<td>Adverb</td>
</tr>
<tr>
<td>AFX</td>
<td>Affix</td>
</tr>
<tr>
<td>CONJ</td>
<td>Conjunction</td>
</tr>
<tr>
<td>DET</td>
<td>Determiner</td>
</tr>
<tr>
<td>INTJ</td>
<td>Interjection</td>
</tr>
<tr>
<td>N</td>
<td>Noun</td>
</tr>
<tr>
<td>NUM</td>
<td>Number or numeric expression</td>
</tr>
<tr>
<td>PPOS</td>
<td>Preposition</td>
</tr>
<tr>
<td>PTCL</td>
<td>Participle</td>
</tr>
<tr>
<td>PRO</td>
<td>Pronoun</td>
</tr>
<tr>
<td>PN</td>
<td>Proper noun</td>
</tr>
<tr>
<td>PUNC</td>
<td>Punctuation</td>
</tr>
<tr>
<td>V</td>
<td>Verb</td>
</tr>
</tbody>
</table>

Noun Group Extraction

Noun groups provide more relevant information than simple nouns. A noun group is defined as a sequence of nouns and their modifiers. Noun group extraction uses part-of-speech tagging to identify nouns and their adjacent noun and adjective modifiers that together form a noun group. Examples of noun groups are “weeklong cruises” and “Middle Eastern languages.”

Entity Identification

Entity identification uses SAS linguistic technologies to classify sequences of words into predefined classes. These classes are assigned as roles for the corresponding sequences. For example, “nlpPerson,” “nlpPlace,” “nlpOrganization,” and “nlpMeasure” are identified as classes for “George W. Bush,” “Boston,” “SAS Institute,” “2.5 inches,” respectively. Table 3.6 shows all valid entities for English. Not all languages support all entities. Table 3.7 and Table 3.8 indicate the languages that are available for each entity.

Table 3.6 All Valid English Entities

<table>
<thead>
<tr>
<th>Entities</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlpDate</td>
<td>Date</td>
</tr>
<tr>
<td>nlpMeasure</td>
<td>Measurement or measurement expression</td>
</tr>
<tr>
<td>nlpMoney</td>
<td>Currency or currency expression</td>
</tr>
<tr>
<td>nlpNounGroup</td>
<td>Phrases that contain multiple words</td>
</tr>
<tr>
<td>nlpOrganization</td>
<td>Organization or company name</td>
</tr>
<tr>
<td>nlpPercent</td>
<td>Percentage or percentage expression</td>
</tr>
<tr>
<td>nlpPerson</td>
<td>Person’s name</td>
</tr>
<tr>
<td>nlpPlace</td>
<td>Addresses, cities, states, and other locations</td>
</tr>
<tr>
<td>nlpTime</td>
<td>Time or time expression</td>
</tr>
</tbody>
</table>
Table 3.7  Supported Language-Entity Pairs, Part 1

<table>
<thead>
<tr>
<th>Language</th>
<th>nlpDate</th>
<th>nlpMeasure</th>
<th>nlpMoney</th>
<th>nlpNounGroup</th>
<th>nlpOrganization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Chinese</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Croatian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Czech</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Danish</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dutch</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>English</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Farsi</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Finnish</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>French</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>German</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Greek</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hebrew</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hindi</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hungarian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Indonesian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Italian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Japanese</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Korean</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Norwegian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Polish</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Portuguese</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Romanian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Russian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Slovak</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Slovene</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Spanish</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Swedish</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Tagalog</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Thai</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Turkish</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 3.8  Supported Language-Entity Pairs, Part 2

<table>
<thead>
<tr>
<th>Language</th>
<th>nlpPercent</th>
<th>nlpPerson</th>
<th>nlpPlace</th>
<th>nlpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Chinese</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Croatian</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Czech</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Danish</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Table 3.8  continued

<table>
<thead>
<tr>
<th>Language</th>
<th>nlpPercent</th>
<th>nlpPerson</th>
<th>nlpPlace</th>
<th>nlpTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dutch</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>English</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Farsi</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Finnish</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>French</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>German</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Greek</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Hebrew</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Hindi</td>
<td>✔</td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Hungarian</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Indonesian</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Italian</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Japanese</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Korean</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Norwegian</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Polish</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Portuguese</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Romanian</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Russian</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Slovak</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Slovene</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Spanish</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Swedish</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Tagalog</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Thai</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Turkish</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Vietnamese</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

Multiword Terms Handling

By default, SAS linguistic technologies tokenize the text to individual words and operate at the word level. Multiword terms provide a control that enables you to specify sequences of words to be interpreted as individual units. For example, “greater than,” “in spite of,” and “as well as” can be defined as multiword terms.

Language Support

Languages supported in the current release are Arabic, Chinese, Croatian, Czech, Danish, Dutch, English, Finnish, French, German, Greek, Hebrew, Hindi, Hungarian, Indonesian, Italian, Japanese, Kazakh, Korean, Norwegian, Polish, Portuguese, Romanian, Russian, Slovak, Slovene, Spanish, Swedish, Tagalog, Thai, Turkish and Vietnamese. By turning off some of the advanced parsing functionality, you might be able to use PROC TEXTMINE effectively with other space-delimited languages.
Term and Cell Weighting

The TERMWGT= option and the CELLEWGT= option control how to weight the frequencies in the compressed term-by-document matrix. The term weight is a positive number that is assigned to each term based on the distribution of that term in the document collection. This weight can be interpreted as an indication of the importance of that term to the document collection. The cell weight is a function that is applied to every entry in the term-by-document matrix; it moderates the effect of a term that is repeated within a document.

Let $f_{i,j}$ be the entry in the $i$th row and $j$th column of the term-by-document matrix, which indicates the time of appearance of term $i$ in document $j$. Assuming that the term weight of term $i$ is $w_i$ and the cell weight function is $g(x)$, the weighted frequency of each entry in the term-by-document matrix is given by $w_i \times g(f_{i,j})$.

When the CELLEWGT=LOG option is specified, the following equation is used to weight cells:

$$g(x) = \log_2(f_{i,j} + 1)$$

The equation reduces the influence of highly frequent terms by applying the log function.

When the TERMWGT=ENTROPY option is specified, the following equation is used to weight terms:

$$w_i = 1 + \sum_j p_{i,j} \log_2(p_{i,j}) \log_2(n)$$

In this equation, $n$ is the number of documents, and $p_{i,j}$ is the probability that term $i$ appears in document $j$, which can be estimated by $p_{i,j} = \frac{f_{i,j}}{g_i}$, where $g_i$ is the global term frequency for term $i$.

When the TERMWGT=MI option is specified, the following equation is used to weight terms:

$$w_i = \max_k \left( \log\left( \frac{P(t_i, C_k)}{P(t_i) P(C_k)} \right) \right)$$

In this equation, $C_k$ is the set of documents that belong to category $k$, $P(C_k)$ is the percentage of documents that belong to category $k$, and $P(t_i, C_k)$ is the percentage of documents that contain term $t_i$ and belong to category $k$. Let $d_i$ be the number of documents that term $i$ appears in. Then $P(t_i) = \frac{d_i}{n}$.

Sparse Format

A matrix is sparse when most of its elements are 0. The term-by-document matrix that the TEXTMINE procedure generates is a sparse matrix. To save storage space, the TEXTMINE procedure supports the COO format for storing a sparse matrix.
Coordinate List (COO) Format

The COO is also known as the transactional format. In this format, the matrix is represented as a set of triples \((i, j, x)\), where \(x\) is an entry in the matrix and \(i\) and \(j\) denote its row and column indices, respectively. When the transactional style is used, all 0 entries in the matrix are ignored in the output, thereby saving storing space when the matrix is sparse. The COO format is good for incremental matrix construction. For example, it is easy to add new rows and new columns to the matrix by inserting more tuples in the list.

Singular Value Decomposition

Singular value decomposition (SVD) of a matrix \(A\) factors \(A\) into three matrices such that \(A = U \Sigma V^\top\). Singular value decomposition also requires that the columns of \(U\) and \(V\) be orthogonal and that \(\Sigma\) be a real-valued diagonal matrix that contains monotonically decreasing, nonnegative entries. The entries of \(\Sigma\) are called singular values. The columns of \(U\) and \(V\) are called left and right singular vectors, respectively. A truncated singular value decomposition calculates only the first \(k\) singular values and their corresponding left and right singular vectors. In information retrieval, singular value decomposition of a term-by-document matrix is also known as latent semantic indexing (LSI).

Applications in Text Mining

Let \(A \in \mathbb{R}^{m \times n}\) be a term-by-document matrix, where \(m\) is the number of terms and \(n\) is the number of documents. The SVD statement has two main functions: to calculate a truncated singular value decomposition (SVD) of \(A\), and to project the columns of \(A\) onto the left singular vectors to generate a new representation of the documents that has a much lower dimensionality. The output of the SVD statement is a truncated singular value decomposition of \(A\), for which the parameter \(k\) defines how many singular values and singular vectors to compute. Singular value decomposition reduces the dimension of the term-by-document matrix and reveals themes that are present in the document collection.

In general, the value of \(k\) must be large enough to capture the meaning of the document collection, yet small enough to ignore the noise. You can specify this value explicitly in the K= option or accept a value that is recommended by the TEXTMINE procedure. A value between 50 and 200 should work well for a document collection that contains thousands of documents.

An important purpose of singular value decomposition is to reduce a high-dimensional term-by-document matrix into a low-dimensional representation that reveals information about the document collection. The columns of the \(A\) form the coordinates of the document space, and the rows form the coordinates of the term space. Each document in the collection is represented as a vector in \(m\)-dimensional space and each term as a vector in \(n\)-dimensional space. The singular value decomposition captures this same information by using a smaller number of basis vectors than would be necessary if you analyzed \(A\) directly.

For example, consider the columns of \(A\), which represent the document space. By construction, the columns of \(U\) also reside in \(m\)-dimensional space. If \(U\) has only one column, the line between that vector and the origin would form the best fit line, in a least squares sense, to the original document space. If \(U\) has two columns, then these columns would form the best fit plane to the original document space. In general, the first \(k\) columns of \(U\) form the best fit \(k\)-dimensional subspace for the document space. Thus, you can project the columns of \(A\) onto the first \(k\) columns of \(U\) in order to optimally reduce the dimension of the document space from \(m\) to \(k\).

The projection of a document \(d\) (one column of \(A\)) onto \(U\) results in \(k\) real numbers that are defined by the inner product \(d\) with each column of \(U\). That is, \(p_i = d^\top u_i\). With this representation, each document forms
a $k$-dimensional vector that can be considered a theme in the document collection. You can then calculate the Euclidean distance between each document and each column of $U$ to determine the documents that are described by this theme.

In a similar fashion, you can repeat the previous process by using the rows of $A$ and the first $k$ columns of $V$. This generates a best fit $k$-dimensional subspace for the term space. This representation is used to group terms into similar clusters. These clusters also represent concepts that are prevalent in the document collection. Thus, singular value decomposition can be used to cluster both the terms and the documents into meaningful representations of the entire document collection.

**Computation**

The computation of the singular vector decomposition is fully parallelized in PROC TEXTMINE via multithreading and distributed computing. Computing singular value decomposition is an iterative process that involves considerable communication among the computer nodes in a distributed computing environment. Therefore, adding more computer nodes for computing singular value decomposition might not always improve efficiency. Conversely, when the data size is not large enough, adding too many computer nodes for computation might lead to a noticeable increase in communication time and sometimes might even slow down the overall computation.

**SVD-Only Mode**

If you run PROC TEXTMINE without a PARSE statement (called SVD-only mode), PROC TEXTMINE directly takes the term-by-document matrix as input and computes singular value decomposition (SVD). This functionality enables you to parse documents and compute the SVD separately in two procedure calls. This approach is useful when you want to try different parameters for SVD computation after document parsing. When you run PROC TEXTMINE in SVD-only mode, the DATA= option in the PROC TEXTMINE statement names the data table that contains the term-by-document matrix.

**Topic Discovery**

You can use the TEXTMINE procedure to discover topics that exist in your collection. In PROC TEXTMINE, topics are calculated as a “rotation” of the SVD dimensions in order to maximize the sum of squares of the term loadings in the $V$ matrix. This rotation preserves the spatial information that the SVD provides, but it also allows the newly rotated SVD dimensions to become semantically interpretable. Topics are characterized by a set of weighted terms. Documents that contain many of these weighted terms are highly associated with the topic, and documents that contain few of them are less associated with the topic. The term scores are found in the $U$ matrix that has been rotated to maximize the explanatory power of each topic. The columns of the $V$ matrix characterize the strength of the association of each document with each topic. Finally, the TEXTMINE procedure can output a topic table that contains the best set of descriptor terms for each topic.

Because topic discovery is derived from the $U$ matrix of SVD (each column of the $U$ matrix is rotated and corresponds to a topic), topic discovery options are specified in the SVD statement.
Output Data Tables

This section describes the output data tables that PROC TEXTMINE produces when you specify the corresponding option.

The OUCHILD= Data Table

The OUCHILD= option in the PARSE statement specifies the data table to contain a compressed representation of the sparse term-by-document matrix, which is usually very sparse. To save space, this matrix is stored in COO format.

If you do not specify the SHOWDROPPEDTERMS option in the PARSE statement, this data table saves only the kept terms.\(^1\)

The child frequencies are not attributed to their corresponding parent (as they are in the data table specified in the OUTPARENT= option). Using the example in the previous section, the data table that is generated by the OUCHILD= option will have two entries:

\[
\begin{align*}
t1 & \quad d1 & \quad 8 \\
t2 & \quad d1 & \quad 1
\end{align*}
\]

The term count of “said” in \(d1\) is not attributed to its parent, “say.” The data table that is specified in the OUCHILD= option can be combined with the data table that is specified in the OUTTERMS= option to construct the data table that is specified in the OUTPARENT= option.

When you specify the SHOWDROPPEDTERMS option in the PARSE statement, the data table saves all the terms that appear in the data table that is specified in the OUTPARENT= option in the PARSE statement.

The OUTCONFIG= Data Table

The OUTCONFIG= option in the PARSE statement specifies a SAS data table to contain the configuration that PROC TEXTMINE uses in the current run. The primary purpose of this data table is to relay the configuration information from the TEXTMINE procedure to the TMSCORE procedure so that the TMSCORE procedure can use options that are consistent with the TEXTMINE procedure during scoring.

Table 3.9 shows the configuration information that is contained in this data table.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Indicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>Source language of the documents</td>
</tr>
<tr>
<td>Stemming</td>
<td>Whether stemming is used: “Y” indicates that stemming is used, and “N” indicates that it is not used</td>
</tr>
<tr>
<td>Tagging</td>
<td>Whether tagging is used: “Y” indicates that tagging is used, and “N” indicates that it is not used</td>
</tr>
<tr>
<td>NG</td>
<td>Whether noun grouping is used: “Y” indicates that noun grouping is used, and “N” indicates that it is not used</td>
</tr>
</tbody>
</table>

\(^1\)Kept terms are terms that are marked as kept in the data table specified in the OUTTERMS= option in the PARSE statement.
Table 3.9  continued

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities</td>
<td>Whether entities should be extracted: “STD” indicates that entities should be extracted, and “N” indicates that entities should not be extracted. When the SELECT statement is specified, “K” indicates that entities are kept, and “D” indicates that entities are ignored.</td>
</tr>
<tr>
<td>Multiterm</td>
<td>The name of the multiterm SAS data table</td>
</tr>
<tr>
<td>Cellwgt</td>
<td>How the cells of the term-by-document matrix are weighted</td>
</tr>
</tbody>
</table>

The contents of this data table are case-sensitive.

**The OUTDOCPRO= Data Table**

The OUTDOCPRO= option in the SVD statement specifies a SAS data table to contain the projections of the columns of the term-by-document matrix onto the columns of U. Because each column of the term-by-document matrix corresponds to a document, the output forms a new representation of the input documents in a space that has much lower dimensionality. If the K= option in the SVD statement is set to k and the input data table contains n documents, the output will have n rows and k + 1 columns. Each row of the output corresponds to a document. The first column of the output contains the ID of the documents, and the name of the column is the same as the variable that is specified in the DOC_ID statement. The remaining k columns are the projections and are named “COL1” to “COLk.”

**The OUTPARENT= Data Table**

The OUTPARENT= option in the PARSE statement specifies a SAS data table to contain a compressed representation of the sparse term-by-document matrix. The term-by-document matrix is usually very sparse.² To save space, this matrix is stored in COO format.

This data table contains three columns: _TERMNUM_, _DOCUMENT_, and _COUNT_. The _TERMNUM_ column contains the ID of the terms (which corresponds to the “Key” column of the data table that is generated by the OUTTERMS= option), the _DOCUMENT_ column contains the ID of the documents, and the _COUNT_ column contains the term counts. For example, (t1 d1 k) means that term t1 appears k times in document d1.

The term counts can be weighted, if requested. The data table saves only the terms that are marked as kept in the data table that is specified in the OUTTERMS= option in the PARSE statement. In the data table, the child frequencies are attributed to the corresponding parent. For example, assume that “said” has term ID t1 and appears eight times in document d1, “say” has term ID t2 and appears one time in document d1, “say” is the parent of “said”, and neither cell weighting nor term weighting is applied. Then the data table that is specified in the OUTPARENT= option will contain the following entry:

```
t2  d1  9
```

The term count of “said” in d1 is attributed to its parent, “say.”

---

²Many elements of the matrix are 0.
The OUTPOS= Data Table

The OUTPOS= option in the PARSE statement specifies a SAS data table to contain the position information about the child terms’ occurrences in the document collection. Table 3.10 shows the variables in this data table.

### Table 3.10 Variables in the OUTPOS= Data Table

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term</td>
<td>A lowercase version of the term</td>
</tr>
<tr>
<td>Role</td>
<td>The term’s part of speech (this variable is empty if the NOTAGGING option is specified in the PARSE statement)</td>
</tr>
<tr>
<td>Parent</td>
<td>A lowercase version of the parent term</td>
</tr>
<tr>
<td><em>Start</em></td>
<td>The starting position of the term’s occurrence (the first position is 0)</td>
</tr>
<tr>
<td><em>End</em></td>
<td>The ending position of the term’s occurrence</td>
</tr>
<tr>
<td>Sentence</td>
<td>The sentence where the occurrence appears</td>
</tr>
<tr>
<td>Paragraph</td>
<td>The paragraph where the occurrence appears (this has not been implemented in the current release, and the value is always set to 0)</td>
</tr>
<tr>
<td>Document</td>
<td>The ID of the document where the occurrence appears</td>
</tr>
<tr>
<td>Target</td>
<td>The value of the target variable that is associated with the document ID if a variable is specified in the TARGET statement</td>
</tr>
</tbody>
</table>

If you exclude terms by specifying the IGNORE option in the SELECT statement, then those terms are excluded from the OUTPOS= data table. No synonym lists, start lists, or stop lists are used when generating the OUTPOS= data table.

The OUTTERMS= Data Table

The OUTTERMS= option in the PARSE statement specifies a SAS data table to contain the summary information about the terms in the document collection. Table 3.11 shows the variables in this data table.

### Table 3.11 Variables in the OUTTERMS= Data Table

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term</td>
<td>A lowercase version of the term</td>
</tr>
<tr>
<td>Role</td>
<td>The term’s part of speech (this variable is empty if the NOTAGGING option is specified in the PARSE statement)</td>
</tr>
<tr>
<td>Attribute</td>
<td>An indication of the characters that compose the term. Possible attributes are as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Alpha</td>
<td>only alphabetic characters</td>
</tr>
<tr>
<td>Mixed</td>
<td>a combination of attributes</td>
</tr>
<tr>
<td>Num</td>
<td>only numbers</td>
</tr>
<tr>
<td>Punct</td>
<td>punctuation characters</td>
</tr>
<tr>
<td>Entity</td>
<td>an identified entity</td>
</tr>
</tbody>
</table>
Table 3.11  continued

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freq</td>
<td>The frequency of a term in the entire document collection</td>
</tr>
<tr>
<td>Numdocs</td>
<td>The number of documents that contain the term</td>
</tr>
<tr>
<td>_keep</td>
<td>The keep status of the term: “Y” indicates that the term is kept for analysis, and “N” indicates that the term should be dropped in later stages of analysis. To ensure that the OUTTERMS= data table is of a reasonable size, only terms that have _keep=Y are kept in the OUTTERMS= data table by default.</td>
</tr>
<tr>
<td>Key</td>
<td>The assigned term number (each unique term in the parsed documents and each unique parent term has a unique Key value)</td>
</tr>
<tr>
<td>Parent</td>
<td>The Key value of the term’s parent or a “.” (period):</td>
</tr>
<tr>
<td></td>
<td>• If a term has a parent, this variable contains the term number of that parent.</td>
</tr>
<tr>
<td></td>
<td>• If a term does not have a parent, this value is a “.” (period).</td>
</tr>
<tr>
<td></td>
<td>• If the values of Key, Parent, and Parent_id are identical, the parent occurs as itself.</td>
</tr>
<tr>
<td></td>
<td>• If the values of Parent and Parent_id are identical but differ from Key, the observation is a child.</td>
</tr>
<tr>
<td>Parent_id</td>
<td>Another description of the term’s parent: Parent contains the parent’s term number if a term is a child, but Parent_id contains this value for all terms.</td>
</tr>
<tr>
<td>_ispar</td>
<td>An indication of term’s status as a parent, child, or neither:</td>
</tr>
<tr>
<td></td>
<td>• A “+” (plus sign) indicates that the term is a parent.</td>
</tr>
<tr>
<td></td>
<td>• A “.” (period) indicates that the term is a child.</td>
</tr>
<tr>
<td></td>
<td>• A missing value indicates that the term is neither a parent nor a child.</td>
</tr>
<tr>
<td>Weight</td>
<td>The weights of the terms</td>
</tr>
</tbody>
</table>

If you do not specify the SHOWDROPPEDTERMS option in the PARSE statement, this data table saves only the terms that have _keep=Y. This helps ensure that the OUTTERMS= data table is of a reasonable size. When you specify the SHOWDROPPEDTERMS option, the data table also saves terms that have _keep=N.
The OUTTOPICS= Data Table

The OUTTOPICS= option specifies the data table for storing the topics that have been discovered. This data table contains three columns: _topicid, _termCutoff, and _name. If the K= option in the SVD statement is set to \( k \), the _topicid column contains the topic index, which is an integer from 1 to \( k \). The _termCutoff column contains the cutoff value that is recommended in order to determine which terms actually belong to the topic. The weights for the terms and topics are contained in \( V \) matrix, which is stored in the data table that is specified in the SVDV= option in the SVD statement. The _name column contains the generated topic name, which is the descriptive label for each topic and provides a synopsis of the discovered topics. The generated topic name contains the terms that have the highest term loadings after the rotation has been performed. The number of terms that are used in the generated name is determined by the NUMLABELS= option in the SVD statement.

Examples: TEXTMINE Procedure

Example 3.1: Parsing with No Options Turned On

This example parses five documents, which are in a generated data table. The following DATA step generates the five documents:

```sas
/* 1) create data table */
data mycas.CarNominations;
infile datalines delimiter='|' missover;
  length text $70 ;
  input text$ i;
datalines;
  The Ford Taurus is the World Car of the Year. |1
  Hyundai won the award last year. |2
  Toyota sold the Toyota Tacoma in bright green. |3
  The Ford Taurus is sold in all colors except for lime green. |4
  The Honda Insight was World Car of the Year in 2008. |5
;
run;
```

The following statements run PROC TEXTMINE to parse the documents.

```sas
/* 2) starting code */
proc textmine data=mycas.CarNominations;
doc_id i;
var text;
parse
  nostemming notagging nonoungroups
  termwgt = none
  cellwgt = none
  reducef = 1
  entities = none
  outparent = mycas.outparent
  outterms = mycas.outterms
```

The OUTTOPICS= option specifies the data table for storing the topics that have been discovered. This data table contains three columns: _topicid, _termCutoff, and _name. If the K= option in the SVD statement is set to \( k \), the _topicid column contains the topic index, which is an integer from 1 to \( k \). The _termCutoff column contains the cutoff value that is recommended in order to determine which terms actually belong to the topic. The weights for the terms and topics are contained in \( V \) matrix, which is stored in the data table that is specified in the SVDV= option in the SVD statement. The _name column contains the generated topic name, which is the descriptive label for each topic and provides a synopsis of the discovered topics. The generated topic name contains the terms that have the highest term loadings after the rotation has been performed. The number of terms that are used in the generated name is determined by the NUMLABELS= option in the SVD statement.
Chapter 3: The TEXTMINE Procedure

outchild = mycas.outchild
outconfig = mycas.outconfig;
run;

/* 3) print outterms data table */
data outterms; set mycas.outterms; run;
proc print data=outterms; run;

Output 3.1.1 shows the content of the mycas.outterms data table. In this example, stemming, part-of-speech tagging, and noun group extraction are suppressed and NONE is specified for entity identification, term and cell weighting, and term filtering. No synonym list, multiterm list, or stop list is specified. As a result of this configuration, there is no child term in the mycas.outterms data table. Also, the mycas.outparent data table and the mycas.outchild data table are exactly the same. The TEXTMINE procedure automatically drops punctuation and numbers.

Output 3.1.1  The mycas.outterms Data Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>Term</th>
<th>Role</th>
<th>Attribute</th>
<th>Freq</th>
<th>numdocs</th>
<th>_keep</th>
<th>Key</th>
<th>Parent</th>
<th>Parent_id</th>
<th>_ispar</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>all</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>toyota</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>1</td>
<td>Y</td>
<td>2</td>
<td>.</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ford</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>3</td>
<td>.</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>tacoma</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>4</td>
<td>.</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>year</td>
<td>Alpha</td>
<td></td>
<td>3</td>
<td>3</td>
<td>Y</td>
<td>5</td>
<td>.</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>taurus</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>6</td>
<td>.</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>won</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>7</td>
<td>.</td>
<td>7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>honda</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>8</td>
<td>.</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>bright</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>9</td>
<td>.</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>sold</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>10</td>
<td>.</td>
<td>10</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>colors</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>11</td>
<td>.</td>
<td>11</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>lime</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>12</td>
<td>.</td>
<td>12</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>except</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>13</td>
<td>.</td>
<td>13</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>hyundai</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>14</td>
<td>.</td>
<td>14</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>in</td>
<td>Alpha</td>
<td></td>
<td>3</td>
<td>3</td>
<td>Y</td>
<td>15</td>
<td>.</td>
<td>15</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>is</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>16</td>
<td>.</td>
<td>16</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>for</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>17</td>
<td>.</td>
<td>17</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>world</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>18</td>
<td>.</td>
<td>18</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>green</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>19</td>
<td>.</td>
<td>19</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>the</td>
<td>Alpha</td>
<td></td>
<td>8</td>
<td>5</td>
<td>Y</td>
<td>20</td>
<td>.</td>
<td>20</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>of</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>21</td>
<td>.</td>
<td>21</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>award</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>22</td>
<td>.</td>
<td>22</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>was</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>23</td>
<td>.</td>
<td>23</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>car</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>24</td>
<td>.</td>
<td>24</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>insight</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>25</td>
<td>.</td>
<td>25</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>last</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>26</td>
<td>.</td>
<td>26</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Example 3.2: Parsing with Stemming

This example uses the data table that is generated in Example 3.1. The following statements run PROC TEXTMINE to parse the documents. Because the NOSTEMMING option is not specified in the PARSE statement, words are stemmed (the default).

```sas
/* create data table */
data mycas.CarNominations;
infile datalines delimiter='|' missover;
length text $70 ;
input text$ i;
datalines;
The Ford Taurus is the World Car of the Year. |1
Hyundai won the award last year. |2
Toyota sold the Toyota Tacoma in bright green. |3
The Ford Taurus is sold in all colors except for lime green. |4
The Honda Insight was World Car of the Year in 2008. |5
run;

proc textmine data=mycas.CarNominations;
doc_id i;
var text;
parse
   notagging nonoungroups
   termwgt = none
   cellwgt = none
   reducef = 1
   entities = none
   outparent= mycas.outparent
   outterms = mycas.outterms
   outchild = mycas.outchild
   outconfig= mycas.outconfig
;
run;
data outterms; set mycas.outterms; run;
proc print data = outterms; run;
```

Output 3.2.1 shows the content of the mycas.outterms data table. In this example, words are stemmed. You can see that the term “sold” now stems to the parent term “sell.” Also, the mycas.outparent data table and the mycas.outchild data table are different. The parent term “sell” shows up in mycas.outparent (key=11), but not the child term “sold” (key=27). Only “sold” appears in the mycas.outchild data table, and “sell” does not appear.
### Output 3.2.1 The mycas.outterms Data Table with Stemming

<table>
<thead>
<tr>
<th>Obs</th>
<th>Term</th>
<th>Role</th>
<th>Attribute</th>
<th>Freq numdocs</th>
<th><em>keep</em></th>
<th>Key</th>
<th>Parent</th>
<th>Parent_id</th>
<th><em>ispar</em></th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>all</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>win</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>2 +</td>
</tr>
<tr>
<td>3</td>
<td>toyota</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>Y</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>ford</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>Y</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>tacoma</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>year</td>
<td>Alpha</td>
<td></td>
<td>3</td>
<td>Y</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>taurus</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>Y</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>won</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>honda</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>bright</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>be</td>
<td>Alpha</td>
<td></td>
<td>3</td>
<td>Y</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>10 +</td>
</tr>
<tr>
<td>12</td>
<td>sold</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>Y</td>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>sell</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>Y</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>colors</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>28</td>
<td>23</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>lime</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>except</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>hyundai</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>in</td>
<td>Alpha</td>
<td></td>
<td>3</td>
<td>Y</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>is</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>Y</td>
<td>29</td>
<td>10</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>for</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>world</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>Y</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>green</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>Y</td>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>the</td>
<td>Alpha</td>
<td></td>
<td>8</td>
<td>Y</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>of</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>Y</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>award</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>was</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>30</td>
<td>10</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>car</td>
<td>Alpha</td>
<td></td>
<td>2</td>
<td>Y</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>color</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>insight</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>last</td>
<td>Alpha</td>
<td></td>
<td>1</td>
<td>Y</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Example 3.3: Adding Entities and Noun Groups

This example uses the data table that is generated in Example 3.1. The following statements run PROC TEXTMINE to parse the documents. Because the NONOUNGROUPS option is not specified in the PARSE statement, noun groups are extracted, and because the ENTITIES=STD option is specified, entities are identified.

```sas
/* create data table */
data mycas.CarNominations;
infile datalines delimiter='|' missover;
length text $70 ;
input text$ i;
datalines;
  The Ford Taurus is the World Car of the Year. |1
  Hyundai won the award last year. |2
  Toyota sold the Toyota Tacoma in bright green. |3
  The Ford Taurus is sold in all colors except for lime green. |4
  The Honda Insight was World Car of the Year in 2008. |5
; run;

proc textmine data=mycas.CarNominations;
doc_id i;
var text i;
PARSE
  notagging
  termwgt = none
  cellwgt = none
  reducef = 1
  entities = std
  outparent = mycas.outparent
  outterms = mycas.outterms
  outchild = mycas.outchild
  outconfig = mycas.outconfig
; run;
data outterms; set mycas.outterms; run;
proc print data=outterms; run;
```

Output 3.3.1 shows the content of the mycas.outterms data table. Compared to Output 3.2.1, the mycas.outterms data table is longer, because it contains entities and noun groups.
## Output 3.3.1 The mycas.outerms Data Table with Noun Group Extraction and Entity Identification

<table>
<thead>
<tr>
<th>Obs</th>
<th>Term</th>
<th>Role</th>
<th>Attribute</th>
<th>Freq</th>
<th>numdocs_keep</th>
<th>Key</th>
<th>Parent</th>
<th>Parent_id_ispar</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>all</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>.</td>
<td>1</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>win</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>.</td>
<td>2</td>
<td>+</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>tacoma</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>3</td>
<td>.</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>year</td>
<td>Alpha</td>
<td>3</td>
<td>3</td>
<td>Y</td>
<td>4</td>
<td>.</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>taurus</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>5</td>
<td>.</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>lime green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>6</td>
<td>.</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>won</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>30</td>
<td>2</td>
<td>.</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>honda</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>7</td>
<td>.</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>bright</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>8</td>
<td>.</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>in 2008</td>
<td>nlpDate</td>
<td>Entity</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>9</td>
<td>.</td>
<td>9</td>
</tr>
<tr>
<td>11</td>
<td>be</td>
<td>Alpha</td>
<td>3</td>
<td>3</td>
<td>Y</td>
<td>10</td>
<td>.</td>
<td>10</td>
<td>+</td>
</tr>
<tr>
<td>12</td>
<td>sold</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>10</td>
<td>12</td>
<td>.</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>bright green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>11</td>
<td>.</td>
<td>11</td>
</tr>
<tr>
<td>14</td>
<td>sell</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>12</td>
<td>.</td>
<td>12</td>
<td>+</td>
</tr>
<tr>
<td>15</td>
<td>colors</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>27</td>
<td>27</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>lime</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>13</td>
<td>.</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>hyundai</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>14</td>
<td>.</td>
<td>14</td>
</tr>
<tr>
<td>18</td>
<td>except</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>15</td>
<td>.</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>in</td>
<td>Alpha</td>
<td>3</td>
<td>3</td>
<td>Y</td>
<td>16</td>
<td>.</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>is</td>
<td>Alpha</td>
<td>2</td>
<td>1</td>
<td>Y</td>
<td>17</td>
<td>.</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>toyota</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>1</td>
<td>Y</td>
<td>18</td>
<td>.</td>
<td>18</td>
</tr>
<tr>
<td>22</td>
<td>last year</td>
<td>nlpDate</td>
<td>Entity</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>19</td>
<td>.</td>
<td>19</td>
</tr>
<tr>
<td>23</td>
<td>ford</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>19</td>
<td>.</td>
<td>19</td>
</tr>
<tr>
<td>24</td>
<td>for</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>20</td>
<td>.</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>world</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>21</td>
<td>.</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>green</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>22</td>
<td>.</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>the</td>
<td>Alpha</td>
<td>8</td>
<td>5</td>
<td>Y</td>
<td>23</td>
<td>.</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>of</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>24</td>
<td>.</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>award</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>25</td>
<td>.</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>was</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>34</td>
<td>10</td>
<td>.</td>
<td>10</td>
</tr>
<tr>
<td>31</td>
<td>car</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>26</td>
<td>.</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>color</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>27</td>
<td>.</td>
<td>27</td>
<td>+</td>
</tr>
<tr>
<td>33</td>
<td>insight</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>28</td>
<td>.</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>last</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>29</td>
<td>.</td>
<td>29</td>
<td>1</td>
</tr>
</tbody>
</table>
Example 3.4: Adding Part-of-Speech Tagging

This example uses the data table that is generated in Example 3.1. The following statements run PROC TEXTMINE to parse the documents. Because the NOTAGGING option is not specified in the PARSE statement, PROC TEXTMINE uses context clues to determine a term’s part of speech.

```sas
/* create data table */
data mycas.CarNominations;
  infile datalines delimiter='|' missover;
  length text $70 ;
  input text$ i;
  datalines;
  The Ford Taurus is the World Car of the Year. |1
  Hyundai won the award last year. |2
  Toyota sold the Toyota Tacoma in bright green. |3
  The Ford Taurus is sold in all colors except for lime green. |4
  The Honda Insight was World Car of the Year in 2008. |5
  ;
  run;

proc textmine data=mycas.CarNominations;
doc_id i;
var text;
parse
  termwgt = none
cellwgt = none
  reducef = 1
  entities = std
  outparent = mycas.outparent
  outterms = mycas.outterms
  outchild = mycas.outchild
  outconfig = mycas.outconfig
;run;
data outterms; set mycas.outterms; run;
proc print data= outterms; run;
```

Output 3.4.1 shows the content of the mycas.outterms data table. Compared to Output 3.3.1, the mycas.outterms data table also contains the part-of-speech tag for the terms.
### Output 3.4.1 The mycas.outterms Data Table with Part-of-Speech Tagging

<table>
<thead>
<tr>
<th>Obs</th>
<th>Term</th>
<th>Role</th>
<th>Attribute</th>
<th>Freq</th>
<th>numdocs</th>
<th>_keep</th>
<th>Key</th>
<th>Parent</th>
<th>Parent_id</th>
<th>_ispar</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>is</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>30</td>
<td>26</td>
<td>26</td>
<td>.</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>was</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>31</td>
<td>26</td>
<td>26</td>
<td>.</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>bright</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>1</td>
<td>.</td>
<td>1</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>taurus</td>
<td>N</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>2</td>
<td>.</td>
<td>2</td>
<td>.</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>won</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>32</td>
<td>17</td>
<td>17</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>for</td>
<td>PPOS</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>3</td>
<td>.</td>
<td>3</td>
<td>.</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>lime green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>4</td>
<td>.</td>
<td>4</td>
<td>.</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>lime</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>5</td>
<td>.</td>
<td>5</td>
<td>.</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>in 2008</td>
<td>nlpDate</td>
<td>Entity</td>
<td>1</td>
<td>1 Y</td>
<td>6</td>
<td>.</td>
<td>6</td>
<td>.</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>the</td>
<td>DET</td>
<td>Alpha</td>
<td>8</td>
<td>5 Y</td>
<td>7</td>
<td>.</td>
<td>7</td>
<td>.</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>except</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>8</td>
<td>.</td>
<td>8</td>
<td>.</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>bright green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>9</td>
<td>.</td>
<td>9</td>
<td>.</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>color</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>10</td>
<td>.</td>
<td>10</td>
<td>+</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>in</td>
<td>PPOS</td>
<td>Alpha</td>
<td>3</td>
<td>3 Y</td>
<td>11</td>
<td>.</td>
<td>11</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>hyundai</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>1</td>
<td>1 Y</td>
<td>12</td>
<td>.</td>
<td>12</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>sold</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>33</td>
<td>28</td>
<td>28</td>
<td>.</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>toyota</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>1 Y</td>
<td>13</td>
<td>.</td>
<td>13</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>last year</td>
<td>nlpDate</td>
<td>Entity</td>
<td>1</td>
<td>1 Y</td>
<td>14</td>
<td>.</td>
<td>14</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>ford</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>2 Y</td>
<td>15</td>
<td>.</td>
<td>15</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>all</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>16</td>
<td>.</td>
<td>16</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>win</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>17</td>
<td>.</td>
<td>17</td>
<td>+</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>car</td>
<td>PN</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>18</td>
<td>.</td>
<td>18</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>colors</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>34</td>
<td>10</td>
<td>10</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>award</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>19</td>
<td>.</td>
<td>19</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>insight</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>20</td>
<td>.</td>
<td>20</td>
<td>.</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>of</td>
<td>PPOS</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>21</td>
<td>.</td>
<td>21</td>
<td>.</td>
<td>21</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>honda</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>22</td>
<td>.</td>
<td>22</td>
<td>.</td>
<td>22</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>world</td>
<td>PN</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>23</td>
<td>.</td>
<td>23</td>
<td>.</td>
<td>23</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>last</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>24</td>
<td>.</td>
<td>24</td>
<td>.</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>green</td>
<td>N</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>25</td>
<td>.</td>
<td>25</td>
<td>.</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>be</td>
<td>V</td>
<td>Alpha</td>
<td>3</td>
<td>3 Y</td>
<td>26</td>
<td>.</td>
<td>26</td>
<td>+</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>tacoma</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>27</td>
<td>.</td>
<td>27</td>
<td>.</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>sell</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>28</td>
<td>.</td>
<td>28</td>
<td>+</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>year</td>
<td>N</td>
<td>Alpha</td>
<td>3</td>
<td>3 Y</td>
<td>29</td>
<td>.</td>
<td>29</td>
<td>.</td>
<td>29</td>
<td>1</td>
</tr>
</tbody>
</table>
Example 3.5: Adding Synonyms

This example uses the data table that is generated in Example 3.1. So far, by looking at the mycas.outterms data tables that are generated by Example 3.1 to Example 3.4, you can see that the data are very “vehicle focused.” If what is important to you is whether or not a car is mentioned in the text, and not the particular model, then you can use a synonym list to map each vehicle model to the broader term “car”. The following DATA step generates the synonym list, and the following statements show this mapping:

```sas
/* create data table */
data mycas.CarNominations;
infile datalines delimiter='|' missover;
length text $70;
input text$ i;
datalines;
The Ford Taurus is the World Car of the Year. |1
Hyundai won the award last year. |2
Toyota sold the Toyota Tacoma in bright green. |3
The Ford Taurus is sold in all colors except for lime green. |4
The Honda Insight was World Car of the Year in 2008. |5
; run;
/* create synonym list */
data mycas.synds;
infile datalines delimiter=',';
length Term $13;
input Term $ TermRole $ Parent $ ParentRole$;
datalines;
insight, PN, car, N,
taurus, N, car, N,
tacoma, PN, car, N,
;
run;

proc textmine data=mycas.CarNominations;
doc_id i;
var text;
parse
termwgt = none
cellwgt = none
reducef = 1
entities = std
synonym = mycas.synds
outparent = mycas.outparent
outterms = mycas.outterms
outchild = mycas.outchild
outconfig = mycas.outconfig
;
RUN;
```

```sas
data outterms; set mycas.outterms; run;
proc print data=outterms; run;
```
Output 3.5.1 shows the content of the mycas.outterms data table. You can see that the term “insight” is assigned the parent term “car”. Only the term “car” appears in the mycas.outparent data table.

**Output 3.5.1** The mycas.outterms Data Table with Synonym Mapping

<table>
<thead>
<tr>
<th>Obs</th>
<th>Term</th>
<th>Role</th>
<th>Attribute</th>
<th>Freq</th>
<th>numdocs</th>
<th>_keep</th>
<th>Key</th>
<th>Parent</th>
<th>Parent_id</th>
<th>_ispar</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>is</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>28</td>
<td>25</td>
<td>25</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>was</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>29</td>
<td>25</td>
<td>25</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>bright</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>1</td>
<td>.</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>taurus</td>
<td>N</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>30</td>
<td>2</td>
<td>2</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>car</td>
<td>N</td>
<td>Alpha</td>
<td>4</td>
<td>4</td>
<td>Y</td>
<td>2</td>
<td>.</td>
<td>2 +</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>won</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>31</td>
<td>17</td>
<td>17</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>for</td>
<td>PPOS</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>3</td>
<td>.</td>
<td>3</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>lime green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>4</td>
<td>.</td>
<td>4</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>lime</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>5</td>
<td>.</td>
<td>5</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>in 2008</td>
<td>nlpDate</td>
<td>Entity</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>6</td>
<td>.</td>
<td>6</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>the</td>
<td>DET</td>
<td>Alpha</td>
<td>8</td>
<td>5</td>
<td>Y</td>
<td>7</td>
<td>.</td>
<td>7</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>except</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>8</td>
<td>.</td>
<td>8</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>bright green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>9</td>
<td>.</td>
<td>9</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>color</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>10</td>
<td>.</td>
<td>10 +</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>in</td>
<td>PPOS</td>
<td>Alpha</td>
<td>3</td>
<td>3</td>
<td>Y</td>
<td>11</td>
<td>.</td>
<td>11</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>hyundai</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>12</td>
<td>.</td>
<td>12</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>sold</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>32</td>
<td>26</td>
<td>26</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>toyota</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>1</td>
<td>Y</td>
<td>13</td>
<td>.</td>
<td>13</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>last year</td>
<td>nlpDate</td>
<td>Entity</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>14</td>
<td>.</td>
<td>14</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>ford</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>15</td>
<td>.</td>
<td>15</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>all</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>16</td>
<td>.</td>
<td>16</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>win</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>17</td>
<td>.</td>
<td>17 +</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>car</td>
<td>PN</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>18</td>
<td>.</td>
<td>18</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>colors</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>33</td>
<td>10</td>
<td>10 +</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>award</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>19</td>
<td>.</td>
<td>19</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>insight</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>34</td>
<td>2</td>
<td>2</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>of</td>
<td>PPOS</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>20</td>
<td>.</td>
<td>20</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>honda</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>21</td>
<td>.</td>
<td>21</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>world</td>
<td>PN</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>22</td>
<td>.</td>
<td>22</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>last</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>23</td>
<td>.</td>
<td>23</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>green</td>
<td>N</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>24</td>
<td>.</td>
<td>24</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>be</td>
<td>V</td>
<td>Alpha</td>
<td>3</td>
<td>3</td>
<td>Y</td>
<td>25</td>
<td>.</td>
<td>25 +</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>tacoma</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>35</td>
<td>2</td>
<td>2</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>sell</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>26</td>
<td>.</td>
<td>26 +</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>year</td>
<td>N</td>
<td>Alpha</td>
<td>3</td>
<td>3</td>
<td>Y</td>
<td>27</td>
<td>.</td>
<td>27</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Example 3.6: Adding a Custom Stop List

This example uses the data table that is generated in Example 3.1 and uses a stop list to drop the term “car” functioning as a proper noun.

```sas
/* create data table */
data mycas.CarNominations;
infile datalines delimiter='|' missover;
length text $70 ;
input text$ i;
datalines;
The Ford Taurus is the World Car of the Year. |1
Hyundai won the award last year. |2
Toyota sold the Toyota Tacoma in bright green. |3
The Ford Taurus is sold in all colors except for lime green. |4
The Honda Insight was World Car of the Year in 2008. |5
;
run;

data mycas.newStopList;
length Term $16 TermRole $16;
infile datalines delimiter=',';
input Term $ TermRole $;
datalines;
   car, PN,
;
run;

proc textmine data=mycas.CarNominations;
doc_id i;
var text;
parse
termwgt = none
cellwgt = none
reducef = 1
entities = std
stop = mycas.newStopList
outparent = mycas.outparent
outterms = mycas.outterms
outchild = mycas.outchild
outconfig = mycas.outconfig
;
run;

data outterms; set mycas.outterms; run;
proc print data= outterms; run;
```

Output 3.6.1 shows the content of the mycas.outterms data table. You can see that the term “car, PN” is not in the mycas.outterms data table because that term and role were added to the custom stop list.
## Output 3.6.1 The mycas.outterms Data Table Filtered Using Stop List

<table>
<thead>
<tr>
<th>Obs</th>
<th>Term</th>
<th>Role</th>
<th>Attribute</th>
<th>Freq</th>
<th>numdocs_keep</th>
<th>Key</th>
<th>Parent</th>
<th>Parent_id</th>
<th>ispar</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>is</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>29</td>
<td>25</td>
<td>25</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>was</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>30</td>
<td>25</td>
<td>25 .</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>bright</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>1</td>
<td>.</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>taurus</td>
<td>N</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>2</td>
<td>.</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>won</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>31</td>
<td>17</td>
<td>17 .</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>for</td>
<td>PPOS</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>3</td>
<td>.</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>lime green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>4</td>
<td>.</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>lime</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>5</td>
<td>.</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>in 2008</td>
<td>nlpDate</td>
<td>Entity</td>
<td>1</td>
<td>1 Y</td>
<td>6</td>
<td>.</td>
<td>6</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>the</td>
<td>DET</td>
<td>Alpha</td>
<td>8</td>
<td>5 Y</td>
<td>7</td>
<td>.</td>
<td>7</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>except</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>8</td>
<td>.</td>
<td>8</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>bright green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>9</td>
<td>.</td>
<td>9</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>color</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>10</td>
<td>.</td>
<td>10 +</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>in</td>
<td>PPOS</td>
<td>Alpha</td>
<td>3</td>
<td>3 Y</td>
<td>11</td>
<td>.</td>
<td>11</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>hyundai</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>1</td>
<td>1 Y</td>
<td>12</td>
<td>.</td>
<td>12</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>sold</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>32</td>
<td>27</td>
<td>27 .</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>toyota</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>1 Y</td>
<td>13</td>
<td>.</td>
<td>13</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>last year</td>
<td>nlpDate</td>
<td>Entity</td>
<td>1</td>
<td>1 Y</td>
<td>14</td>
<td>.</td>
<td>14</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>ford</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>2 Y</td>
<td>15</td>
<td>.</td>
<td>15</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>all</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>16</td>
<td>.</td>
<td>16</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>win</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>17</td>
<td>.</td>
<td>17 +</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>colors</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>33</td>
<td>10</td>
<td>10 .</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>award</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>18</td>
<td>.</td>
<td>18</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>insight</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>19</td>
<td>.</td>
<td>19</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>of</td>
<td>PPOS</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>20</td>
<td>.</td>
<td>20</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>honda</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>21</td>
<td>.</td>
<td>21</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>world</td>
<td>PN</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>22</td>
<td>.</td>
<td>22</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>last</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>23</td>
<td>.</td>
<td>23</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>green</td>
<td>N</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>24</td>
<td>.</td>
<td>24</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>be</td>
<td>V</td>
<td>Alpha</td>
<td>3</td>
<td>3 Y</td>
<td>25</td>
<td>.</td>
<td>25 +</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>tacoma</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>26</td>
<td>.</td>
<td>26</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>sell</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>27</td>
<td>.</td>
<td>27 +</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>year</td>
<td>N</td>
<td>Alpha</td>
<td>3</td>
<td>3 Y</td>
<td>28</td>
<td>.</td>
<td>28</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Example 3.7: Adding a Multiterm List

You can specify a multiterm list to define terms that consist of multiple words. This example uses the data table that is generated in Example 3.1 to show how to use the MULTITERM= option. The following DATA steps generate and uses a multiterm list:

```plaintext
/* create data table */
data mycas.CarNominations;
infile datalines delimiter='|' missover;
length text $70 ;
input text$ i;
datalines;
  The Ford Taurus is the World Car of the Year. |1
  Hyundai won the award last year. |2
  Toyota sold the Toyota Tacoma in bright green. |3
  The Ford Taurus is sold in all colors except for lime green. |4
  The Honda Insight was World Car of the Year in 2008. |5
; run;

/* create multiterm list */
data mycas.multiterms;
infile datalines delimiter='|';
length multiterm $64;
input multiterm$;
datalines;
  except for :3:Prep
;
run;

proc textmine data=mycas.CarNominations;
doc_id i;
var text;
parse
  termwgt = none
  cellwgt = none
  reducef = 1
  entities = std
  multiterm = mycas.multiterms
  outparent = mycas.outparent
  outterms = mycas.outterms
  outchild = mycas.outchild
  outconfig = mycas.outconfig
;
run;

data outterms; set mycas.outterms; run;
proc print data= outterms; run;
```

Output 3.7.1 shows the content of the mycas.outterms data table. In the preceding statements, “except for” is defined as an individual term in the third DATA step. In the mycas.outterms data table, you can see that the two terms “except” and “for” have become one term, “except for.”
## Output 3.7.1 The mycas.outterms Data Table Using a Multiterm List

<table>
<thead>
<tr>
<th>Obs</th>
<th>Term</th>
<th>Role</th>
<th>Attribute</th>
<th>Freq numdocs</th>
<th>_keep</th>
<th>Key</th>
<th>Parent</th>
<th>Parent_id ispar</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>is</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>29</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>was</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>30</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>bright</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>1</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>taurus</td>
<td>N</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>2</td>
<td>.</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>won</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>31</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>lime green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>3</td>
<td>.</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>except for</td>
<td>PPOS</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>4</td>
<td>.</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>lime</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>5</td>
<td>.</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>in 2008</td>
<td>nlpDate</td>
<td>Entity</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>6</td>
<td>.</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>the</td>
<td>DET</td>
<td>Alpha</td>
<td>8</td>
<td>5</td>
<td>Y</td>
<td>7</td>
<td>.</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>bright green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>8</td>
<td>.</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>color</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>9</td>
<td>.</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>in</td>
<td>PPOS</td>
<td>Alpha</td>
<td>3</td>
<td>3</td>
<td>Y</td>
<td>10</td>
<td>.</td>
<td>10</td>
</tr>
<tr>
<td>14</td>
<td>hyundai</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>11</td>
<td>.</td>
<td>11</td>
</tr>
<tr>
<td>15</td>
<td>sold</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>32</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>toyota</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>1</td>
<td>Y</td>
<td>12</td>
<td>.</td>
<td>12</td>
</tr>
<tr>
<td>17</td>
<td>last year</td>
<td>nlpDate</td>
<td>Entity</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>13</td>
<td>.</td>
<td>13</td>
</tr>
<tr>
<td>18</td>
<td>ford</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>14</td>
<td>.</td>
<td>14</td>
</tr>
<tr>
<td>19</td>
<td>all</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>15</td>
<td>.</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>win</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>16</td>
<td>.</td>
<td>16</td>
</tr>
<tr>
<td>21</td>
<td>car</td>
<td>PN</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>17</td>
<td>.</td>
<td>17</td>
</tr>
<tr>
<td>22</td>
<td>colors</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>33</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>23</td>
<td>award</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>18</td>
<td>.</td>
<td>18</td>
</tr>
<tr>
<td>24</td>
<td>insight</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>19</td>
<td>.</td>
<td>19</td>
</tr>
<tr>
<td>25</td>
<td>of</td>
<td>PPOS</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>20</td>
<td>.</td>
<td>20</td>
</tr>
<tr>
<td>26</td>
<td>honda</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>21</td>
<td>.</td>
<td>21</td>
</tr>
<tr>
<td>27</td>
<td>world</td>
<td>PN</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>22</td>
<td>.</td>
<td>22</td>
</tr>
<tr>
<td>28</td>
<td>last</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>23</td>
<td>.</td>
<td>23</td>
</tr>
<tr>
<td>29</td>
<td>green</td>
<td>N</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>24</td>
<td>.</td>
<td>24</td>
</tr>
<tr>
<td>30</td>
<td>be</td>
<td>V</td>
<td>Alpha</td>
<td>3</td>
<td>3</td>
<td>Y</td>
<td>25</td>
<td>.</td>
<td>25</td>
</tr>
<tr>
<td>31</td>
<td>tacoma</td>
<td>PN</td>
<td>Alpha</td>
<td>1</td>
<td>1</td>
<td>Y</td>
<td>26</td>
<td>.</td>
<td>26</td>
</tr>
<tr>
<td>32</td>
<td>sell</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2</td>
<td>Y</td>
<td>27</td>
<td>.</td>
<td>27</td>
</tr>
<tr>
<td>33</td>
<td>year</td>
<td>N</td>
<td>Alpha</td>
<td>3</td>
<td>3</td>
<td>Y</td>
<td>28</td>
<td>.</td>
<td>28</td>
</tr>
</tbody>
</table>
Example 3.8: Selecting Parts of Speech and Entities to Ignore

This example uses the data table that is generated in Example 3.1. If you want to eliminate prepositions, determiners, and proper nouns from your analysis, you can add a SELECT statement that lists these part-of-speech labels. If you also want to eliminate entities that are labeled “nlpDate,” you can add another SELECT statement that includes “nlpDate” in the label list.

```sas
/* create data table */
data mycas.CarNominations;
infile datalines delimiter='|' missover;
length text $70 ;
input text$ i;
datalines;
   The Ford Taurus is the World Car of the Year. |1
   Hyundai won the award last year. |2
   Toyota sold the Toyota Tacoma in bright green. |3
   The Ford Taurus is sold in all colors except for lime green. |4
   The Honda Insight was World Car of the Year in 2008. |5
;
run;

proc textmine data=mycas.CarNominations;
doc_id i;
var text;
parse
   termwgt = none
   cellwgt = none
   reducef = 1
   entities = std
   outparent = mycas.outparent
   outterms = mycas.outterms
   outchild = mycas.outchild
   outconfig = mycas.outconfig
;
   select "PPOS" "DET" "PN"/ignore;
   select "nlpDate"/group="entities" ignore;
run;

data outterms; set mycas.outterms; run;
proc print data= outterms; run;
```

Output 3.8.1 shows the content of the mycas.outterms data table. You can see that prepositions, determiners, and proper nouns are excluded. Terms that are labeled “nlpDate” are also excluded.
## Output 3.8.1

The mycas.outterms Data Table Ignoring Specified Parts of Speech and Entities

<table>
<thead>
<tr>
<th>Obs</th>
<th>Term</th>
<th>Role</th>
<th>Attribute</th>
<th>Freq</th>
<th>numdocs_KEEP</th>
<th>Key</th>
<th>Parent</th>
<th>Parent_id_ispar</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>is</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>19</td>
<td>16</td>
<td>16 .</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>was</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>20</td>
<td>16</td>
<td>16 .</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>bright</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>1</td>
<td>.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>taurus</td>
<td>N</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>won</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>21</td>
<td>12</td>
<td>12 .</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>lime green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>3</td>
<td>.</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>lime</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>4</td>
<td>.</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>except</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>5</td>
<td>.</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>bright green</td>
<td>nlpNounGroup</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>6</td>
<td>.</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>color</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>7</td>
<td>.</td>
<td>7 +</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>hyundai</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>1</td>
<td>1 Y</td>
<td>8</td>
<td>.</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>sold</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>22</td>
<td>17</td>
<td>17 .</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>toyota</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>1 Y</td>
<td>9</td>
<td>.</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>ford</td>
<td>nlpOrganization</td>
<td>Entity</td>
<td>2</td>
<td>2 Y</td>
<td>10</td>
<td>.</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>all</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>11</td>
<td>.</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>win</td>
<td>V</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>12</td>
<td>.</td>
<td>12 +</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>colors</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>23</td>
<td>7</td>
<td>7 .</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>award</td>
<td>N</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>13</td>
<td>.</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>last</td>
<td>A</td>
<td>Alpha</td>
<td>1</td>
<td>1 Y</td>
<td>14</td>
<td>.</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>green</td>
<td>N</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>15</td>
<td>.</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>be</td>
<td>V</td>
<td>Alpha</td>
<td>3</td>
<td>3 Y</td>
<td>16</td>
<td>.</td>
<td>16 +</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>sell</td>
<td>V</td>
<td>Alpha</td>
<td>2</td>
<td>2 Y</td>
<td>17</td>
<td>.</td>
<td>17 +</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>year</td>
<td>N</td>
<td>Alpha</td>
<td>3</td>
<td>3 Y</td>
<td>18</td>
<td>.</td>
<td>18</td>
<td>1</td>
</tr>
</tbody>
</table>
Chapter 4
The TMSCORE Procedure

Overview: TMSCORE Procedure

The TMSCORE procedure scores textual data in SAS Viya. In text mining, scoring is the process of applying parsing and singular value decomposition (SVD) projections to new textual data. The TMSCORE procedure performs this scoring of new documents, and its primary outputs are the Outparent data table (which holds the parsing results of the term-by-document matrix) and the Outdocpro data table (which holds the reduced-dimensional representation of the score collection). PROC TMSCORE uses some of the output data tables of the TEXTMINE procedure as input data to ensure consistency between scoring and training. During scoring, the new textual data must be parsed using the same settings that the training data were parsed with, indexed using only the subset of terms that were used during training, and projected onto the reduced-dimensional subspace of the singular value decomposition that was derived from the training data. To facilitate this process, you specify the CONFIG=, TERMS=, and SVDU= options in PROC TEXTMINE to create three data tables (Outconfig, Outterms, and Svdu, respectively), and then you specify those three data tables as inputs to PROC TMSCORE. For more information about these data tables, see the CONFIG=, TERMS=, and SVDU= options, respectively, in the section “PROC TMSCORE Statement” on page 88.
PROC TMSCORE Features

The TMSCORE procedure processes large-scale textual data in parallel to achieve efficiency and scalability. The following list summarizes the basic features of PROC TMSCORE:

- Functionalities that are related to document parsing, term-by-document matrix creation, and dimension reduction are integrated into one procedure to process data more efficiently.
- Parsing and term-by-document matrix creation are performed in parallel.
- Computation of document projection is performed in parallel.
- All phases of processing use a high degree of multithreading.

Using CAS Sessions and CAS Engine Librefs

SAS Cloud Analytic Services (CAS) is the analytic server and associated cloud services in SAS Viya. This section describes how to create a CAS session and set up a CAS engine libref that you can use to connect to the CAS session. It assumes that you have a CAS server already available; contact your system administrator if you need help starting and terminating a server. This CAS server is identified by specifying the host on which it runs and the port on which it listens for communications. To simplify your interactions with this CAS server, the host information and port information for the server are stored as SAS option values that are retrieved automatically whenever this CAS server needs to be accessed. You can examine the host and port values for the server at your site by using the following statements:

```
proc options option=(CASHOST CASPORT);
run;
```

In addition to starting a CAS server, your system administrator might also have created a CAS session and a CAS engine libref for your use. You can define your own sessions and CAS engine librefs that connect to the CAS server as shown in the following statements:

```
cas mysess;
libname mycas cas sessref=mysess;
```

The CAS statement creates the CAS session named `mysess`, and the LIBNAME statement creates the `mycas` CAS engine libref that you use to connect to this session. It is not necessary to explicitly name the CASHOST and CASPORT of the CAS server in the CAS statement, because these values are retrieved from the corresponding SAS option values.

If you have created the `mysess` session, you can terminate it by using the TERMINATE option in the CAS statement as follows:

```
cas mysess terminate;
```

For more information about the CAS and LIBNAME statements, see the section “Introduction to Shared Concepts” on page 1 in Chapter 1, “Shared Concepts.”
Getting Started: TMSCORE Procedure

**NOTE:** Input data must be in a CAS table that is accessible in your CAS session. You must refer to this table by using a two-level name. The first level must be a CAS engine libref, and the second level must be the table name. For more information, see the sections “Using CAS Sessions and CAS Engine Librefs” on page 1 and “Loading a SAS Data Set onto a CAS Server” on page 2 in Chapter 1, “Shared Concepts.”

The following DATA steps generate two data tables: the mycas.getstart data table contains 36 observations, and the mycas.getstart_score data table contains 31 observations. Both data tables have two variables: the text variable contains the input documents, and the did variable contains the ID of the documents. Each row in each data table represents a “document” for analysis.

```sas
data mycas.getstart;
  infile datalines delimiter='|' missover;
  length text $150;
  input text$ did;
  datalines;
  High-performance analytics hold the key to |1
  unlocking the unprecedented business value of big data.|2
  Organizations looking for optimal ways to gain insights|3
  from big data in shorter reporting windows are turning to SAS.|4
  As the gold-standard leader in business analytics |5
  for more than 36 years,|6
  SAS frees enterprises from the limitations of |7
  traditional computing and enables them |8
  to draw instant benefits from big data.|9
  Faster Time to Insight.|10
  From banking to retail to health care to insurance, |11
  SAS is helping industries glean insights from data |12
  that once took days or weeks in just hours, minutes or seconds.|13
  It’s all about getting to and analyzing relevant data faster.|14
  Revealing previously unseen patterns, sentiments and relationships.|15
  Identifying unknown risks.|16
  And speeding the time to insights.|17
  High-Performance Analytics from SAS Combining industry-leading |18
  analytics software with high-performance computing technologies|19
  produces fast and precise answers to unsolvable problems|20
  and enables our customers to gain greater competitive advantage.|21
  SAS In-Memory Analytics eliminate the need for disk-based processing|22
  allowing for much faster analysis.|23
  SAS In-Database executes analytic logic into the database itself |24
  for improved agility and governance.|25
  SAS Grid Computing creates a centrally managed,|26
  shared environment for processing large jobs|27
  and supporting a growing number of users efficiently.|28
  Together, the components of this integrated, |29
  supercharged platform are changing the decision-making landscape|30
  and redefining how the world solves big data business problems.|31
  Big data is a popular term used to describe the exponential growth,|32
  availability and use of information.|33
  both structured and unstructured.|34
  Much has been written on the big data trend and how it can |35
```
serve as the basis for innovation, differentiation and growth.

```
run;

data mycas.getstart_score;
    infile datalines delimiter='|' missover;
    length text $150;
    input text$ did;
    datalines;
    Big data according to SAS
    At SAS, consider two other dimensions when thinking about big data:
    Variability. In addition to the increasing velocities and varieties of data, data flows can be highly inconsistent with periodic peaks.
    Is something big trending in the social media? Perhaps there is a high-profile IPO looming.
    Maybe swimming with pigs in the Bahamas is suddenly the must-do vacation activity. Daily, seasonal and event-triggered peak data loads can be challenging to manage - especially with social media involved.
    Complexity. When you deal with huge volumes of data, it comes from multiple sources. It is quite an undertaking to link, match, cleanse and transform data across systems. However, it is necessary to connect and correlate relationships, hierarchies and multiple data linkages or your data can quickly spiral out of control. Data governance can help you determine how disparate data relates to common definitions and how to systematically integrate structured and unstructured data assets to produce high-quality information that is useful, appropriate and up-to-date.
    Ultimately, regardless of the factors involved, I believe that the term big data is relative whenever an organization's ability to handle, store and analyze data exceeds its current capacity.
run;
```

The following statements use PROC TEXTMINE for processing the input text data table mycas.getstart and create three data tables (mycas.outconfig, mycas.terms, and mycas.svdu), which can be used in PROC TMSCORE for scoring:

```
proc textmine data = mycas.getstart;
doc_id did;
variables text;
parse
    outterms = mycas.terms
    outconfig = mycas.outconfig
    reducef = 2;
svd
    k = 5
```
Getting Started: TMSCORE Procedure

svdu = mycas.svdu;
run;

The following statements then use PROC TMSCORE to score the input text data table mycas.getstart_score. The statements take the three data tables that are generated by PROC TEXTMINE as input and create a data table named mycas.docpro, which contains the projection of the documents in the input data table mycas.getstart_score.

proc tmscore
data = mycas.getstart_score
terms = mycas.outterms
config = mycas.outconfig
svdu = mycas.svdu
svddocpro = mycas.docpro;
doc_id did;
variables text;
run;

The output from this analysis is presented in Figure 4.1.

The following statements use PROC PRINT to show the content of the first 10 rows of the sorted mycas.docpro data table, which is generated by the TMSCORE procedure:

data docpro;
    set mycas.docpro;
run;
proc sort data=docpro;
by did;
run;
proc print data = docpro (obs=10);
run;

Figure 4.1 shows the output of PROC PRINT.

Figure 4.1 The mycas.docpro Data Table

<table>
<thead>
<tr>
<th>Obs</th>
<th>did</th>
<th>COL1</th>
<th>COL2</th>
<th>COL3</th>
<th>COL4</th>
<th>COL5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.8460041362</td>
<td>-0.022725647</td>
<td>0.1330595299</td>
<td>0.5146460484</td>
<td>0.0345709829</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.3312354984</td>
<td>0.573779031</td>
<td>0.0066225814</td>
<td>0.7472313995</td>
<td>0.0515950108</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0.8520340979</td>
<td>-0.358672789</td>
<td>0.1873407858</td>
<td>0.3187325661</td>
<td>-0.093299024</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0.64928804</td>
<td>0.2747636778</td>
<td>0.4454014167</td>
<td>-0.316447556</td>
<td>0.4521155657</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0.9430684788</td>
<td>-0.185746085</td>
<td>0.0903397136</td>
<td>0.0816038571</td>
<td>0.2475879297</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0.8325586063</td>
<td>-0.174210986</td>
<td>-0.353242685</td>
<td>0.388738294</td>
<td>-0.02447128</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0.901438766</td>
<td>0.0115370594</td>
<td>0.3626555424</td>
<td>0.1689222334</td>
<td>0.1649887393</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>0.6826827301</td>
<td>-0.004113157</td>
<td>-0.213214441</td>
<td>0.3301557457</td>
<td>0.6160066215</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>0.8548352509</td>
<td>0.2464171755</td>
<td>0.3749249411</td>
<td>0.1417168616</td>
<td>0.218821592</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.6727152395</td>
<td>0.1569493092</td>
<td>0.0507091334</td>
<td>-0.653034877</td>
<td>0.306259946</td>
</tr>
</tbody>
</table>
Chapter 4: The TMSCORE Procedure

Syntax: TMSCORE Procedure

The following statements are available in the TMSCORE procedure:

```plaintext
PROC TMSCORE DATA=CAS-libref.data-table < options > ;
VARIABLES variable ;
DOC_ID variable ;
```

PROC TMSCORE Statement

```plaintext
PROC TMSCORE DATA=CAS-libref.data-table < options > ;
```

The PROC TMSCORE statement invokes the procedure. Table 4.1 summarizes the options in the statement by function. The options are then described fully in alphabetical order.

<table>
<thead>
<tr>
<th>option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic Options</td>
<td></td>
</tr>
<tr>
<td>CONFIG=</td>
<td>Specifies the data table that contains the configuration information</td>
</tr>
<tr>
<td>DATA</td>
<td>DOC=</td>
</tr>
<tr>
<td>SVDU=</td>
<td>Specifies the data table that contains the U matrix whose columns are the left singular vectors</td>
</tr>
<tr>
<td>TERMS=</td>
<td>Specifies the data table that contains the terms to be used for scoring</td>
</tr>
<tr>
<td>Output Options</td>
<td></td>
</tr>
<tr>
<td>OUTPARENT=</td>
<td>Specifies the data table that contains the term-by-document frequency matrix that is used to model the document collection. In this matrix, the child terms are not represented and child terms’ frequencies are attributed to their corresponding parents.</td>
</tr>
<tr>
<td>SVDDOCPRO=</td>
<td>Specifies the data table that contains the projections of the documents</td>
</tr>
</tbody>
</table>

You must specify the following option:

- **DATA=** `CAS-libref.data-table` names the input data table for PROC TMSCORE to use. `CAS-libref.data-table` is a two-level name, where `CAS-libref` refers to a collection of information that is defined in the LIBNAME statement and includes the caslib, which includes a path to the data, and a session identifier, which defaults to the active session but which can be explicitly defined in the LIBNAME statement. For more information about `CAS-libref`, see the section “Using CAS Sessions and CAS Engine Librefs” on page 84.
**data-table** specifies the name of the input data table.

The input data table contains documents for PROC TMSCORE to score. Each row of the input data table must contain one text variable and one ID variable, which correspond to the text and the unique ID of a document, respectively.

You can also specify the following options:

- **CONFIG=** `CAS-libref.data-table`
  specifies the input data table that contains configuration information for PROC TMSCORE. `CAS-libref.data-table` is a two-level name, where `CAS-libref` refers to the caslib and session identifier, and `data-table` specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 84. Specify the table that was generated by the OUTCONFIG= option in the PARSE statement of the TEXTMINE procedure during training. For more information about this data table, see the section “The OUTCONFIG= Data Table” on page 63 of Chapter 3, “The TEXTMINE Procedure.”

- **OUTPARENT=** `CAS-libref.data-table`
  specifies the output data table to contain a compressed representation of the sparse term-by-document frequency matrix. `CAS-libref.data-table` is a two-level name, where `CAS-libref` refers to the caslib and session identifier, and `data-table` specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 84. The data table contains only the kept representative terms, and the child frequencies are attributed to the corresponding parent. For more information about the compressed representation of the sparse term-by-document frequency matrix, see the section “The OUTPARENT= Data Table” on page 64 of Chapter 3, “The TEXTMINE Procedure.”

- **SVDDOCPRO=** `CAS-libref.data-table`
  specifies the output data table to contain the reduced dimensional projections for each document. `CAS-libref.data-table` is a two-level name, where `CAS-libref` refers to the caslib and session identifier, and `data-table` specifies the name of the output data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 84. The contents of this data table are formed by multiplying the term-by-document frequency matrix by the input data table that is specified in the SVDU= option and then normalizing the result.

- **SVDU=** `CAS-libref.data-table`
  specifies the input data table that contains the U matrix, which is created during training by PROC TEXTMINE. `CAS-libref.data-table` is a two-level name, where `CAS-libref` refers to the caslib and session identifier, and `data-table` specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 84. The data table contains the information that is needed to project each document into the reduced dimensional space. For more information about the contents of this data table, see the SVDU= option in Chapter 3, “The TEXTMINE Procedure.”

- **TERMS=** `CAS-libref.data-table`
  specifies the input data table of terms to be used by PROC TMSCORE. `CAS-libref.data-table` is a two-level name, where `CAS-libref` refers to the caslib and session identifier, and `data-table` specifies the name of the input data table. For more information about this two-level name, see the DATA= option and the section “Using CAS Sessions and CAS Engine Librefs” on page 84. Specify the table that was generated by the OUTTERMS= option in the PARSE statement of the TEXTMINE procedure during
training. This data table conveys to PROC TMSCORE which terms should be used in the analysis and whether they should be mapped to a parent. The data table also assigns to each term a key that corresponds to the key that is used in the input data table that is specified by the SVDU= option. For more information about this data table, see the section “The OUTTERMS= Data Table” on page 65 of Chapter 3, “The TEXTMINE Procedure.”

---

**DOC_ID Statement**

```
DOC_ID variable ;
```

This statement specifies the `variable` that contains the ID of each document. The ID of each document must be unique; it can be either a number or a string of characters.

---

**VARIABLES Statement**

```
VARIABLES variable ;
VAR variable ;
```

This statement specifies the `variable` that contains the text to be processed.

---

**Details: TMSCORE Procedure**

For information about the techniques that are used for nature language processing, term processing, and singular value decomposition, see the section “Details: TEXTMINE Procedure” on page 56 of Chapter 3, “The TEXTMINE Procedure.”

---

**System Configuration**

**Prerequisites for Running PROC TMSCORE**

To use the TMSCORE procedure, the language binary files that are provided under that license must be available on the grid for parsing text.
Subject Index

options summary
  PARSE statement, 46
  PROC TEXTMINE statement, 44
  PROC TMSCORE statement, 88
  SELECT statement, 51
  SVD statement, 52

sparse matrix
  TEXTMINE procedure, 64

TEXTMINE procedure, 38
  cell weight, 47
  coordinate list (COO) format, 61
  entity, 47
  filtering term by frequency, 49
  input data tables, 45
  language used by input data tables, 45
  multiterm words list, 47
  noun groups, 47
  number of threads, 45
  show dropped terms, 49
  sparse format, 60
  sparse matrix, 64
  start list, 49
  stemming, 48
  stop list, 49
  SVD, singular value decomposition, 61
  synonym list, 49
  tagging, 48
  term weight, 50
  transactional style, 64
  variable name style, 45

TMSCORE procedure, 83
  input data tables, 88
  system configuration, 90

TMSCORE procedure, system configuration
  prerequisite, 90

transactional style
  TEXTMINE procedure, 64
Syntax Index

BOOLRULE procedure, 11
  DOCINFO statement, 14
  OUTPUT statement, 15
  PROC BOOLRULE statement, 12
  SCORE statement, 16
  syntax, 11
  TERMINFO statement, 17
BOOLRULE procedure, DOCINFO statement, 14
  EVENTS= option, 14
  ID= option, 15
  TARGET= option, 15
  TARGETTYPE= option, 15
BOOLRULE procedure, OUTPUT statement, 15
  CANDIDATETERMS= option, 15
  RULES= option, 16
  RULETERMS= option, 16
BOOLRULE procedure, PROC BOOLRULE statement, 12
  DATA= option, 12
  DOC= option, 12
  DOCID= option, 12
  DOCINFO= option, 13
  GNEG= option, 13
  GPOS= option, 13
  MAXCANDIDATES= option, 13
  MAXCANDS= option, 13
  MAXTRIESIN= option, 13
  MAXTRIESOUT= option, 14
  MINSUPPORTS= option, 14
  MNEG= option, 14
  MPOS= option, 14
  TERMID= option, 14
  TERMINFO= option, 14
BOOLRULE procedure, SCORE statement, 16
  OUTMATCH= option, 16
  RULETERMS= option, 16
BOOLRULE procedure, TERMINFO statement, 17
  ID= option, 17
  LABEL= option, 17
CANDIDATETERMS= option
  OUTPUT statement, 15
CELLWT= option
  PARSE statement, 47
COL= option
  SVD statement, 53
CONFIG= option
  TMSCORE statement, 89
DATA= option
  PROC BOOLRULE statement, 12
  PROC TEXTMINE statement, 45
  PROC TMSCORE statement, 88
DOC= option
  PROC BOOLRULE statement, 12
  PROC TMSCORE statement, 88
DOC_ID statement
  TEXTMINE procedure, 45
  TMSCORE procedure, 90
DOCID= option
  PROC BOOLRULE statement, 13
DOCINFO statement
  BOOLRULE procedure, 14
  DOCINFO= option
  PROC BOOLRULE statement, 13
ENTITIES= option
  PARSE statement, 47
ENTRY= option
  SVD statement, 53
EVENTS= option
  DOCINFO statement, 14
EXACTWEIGHT option
  SVD statement, 53
GNEG= option
  PROC BOOLRULE statement, 13
GPOS= option
  PROC BOOLRULE statement, 13
GROUP= option
  SELECT statement, 51
ID= option
  DOCINFO statement, 15
  TERMINFO statement, 17
IGNORE option
  SELECT statement, 51
IN_TERMS= option
  SVD statement, 53
K= option
  SVD statement, 53
KEEP option
  SELECT statement, 51
KEEPVARS, KEEPVARIABLES
  SVD statement, 54
LABEL= option
TERMININFO statement, 17
LABELS option
SELECT statement, 51
LANGUAGE= option
PROC TEXTMINE statement, 45
MAX_K= option
SVD statement, 53
MAXCANDIDATES= option
PROC BOOLRULE statement, 13
MAXCANDS= option
PROC BOOLRULE statement, 13
MAXTRIESIN= option
PROC BOOLRULE statement, 13
MAXTRIESOUT= option
PROC BOOLRULE statement, 14
MINSUPPORTS= option
PROC BOOLRULE statement, 14
MNEG= option
PROC BOOLRULE statement, 14
MPOS= option
PROC BOOLRULE statement, 14
MULTITERM= option
PARSE statement, 47
NEWVARNAMES
TEXTMINE statement, 45
NOCUTOFFS option
SVD statement, 53
NONG option
PARSE statement, 47
NONOUNGROUPS option
PARSE statement, 47
NOSTEMMING option
PARSE statement, 48
NOTAGGING option
PARSE statement, 48
NTHREADS= option
PROC TEXTMINE statement, 45
NUMLABELS= option
SVD statement, 54
OUTCHILD= option
PARSE statement, 48
OUTCONFIG= option
PARSE statement, 48
OUTDOCPRO= option
SVD statement, 54
OUTMATCH= option
SCORE statement, 16
OUTPARENT= option
PARSE statement, 48
PROC TMSCORE statement, 89
OUTPOS= option
PARSE statement, 48
OUTPUT statement
BOOLRULE procedure, 15
OUTTERMS= option
PARSE statement, 49
OUTTOPICS= option
SVD statement, 54
PARSE statement
TEXTMINE procedure, 46
PROC BOOLRULE statement
BOOLRULE procedure, 12
PROC TEXTMINE statement
TEXTMINE procedure, 44
PROC TMSCORE statement
TMSCORE procedure, 88
REDUCEF= option
PARSE statement, 49
RES= option
SVD statement, 54
RESOLUTION= option
SVD statement, 54
ROTATION= option
SVD statement, 55
ROW= option
SVD statement, 55
RSTORE= option
SAVESTATE statement, 50
RULES= option
OUTPUT statement, 16
RULETERMS= option
OUTPUT statement, 16
SCORE statement, 16
SAVESTATE statement
SVMACHINE procedure, 50
SCORE statement
BOOLRULE procedure, 16
SELECT statement
TEXTMINE procedure, 50
SHOWDROPPEDTERMS= option
PARSE statement, 49
START= option
PARSE statement, 49
STOP= option
PARSE statement, 49
SVD statement
TEXTMINE procedure, 52
SVDDOCPRO= option
PROC TMSCORE statement, 89
SVDS= option
SVD statement, 55
SVDU= option
PROC TMSCORE statement, 89
SVD statement, 55
SVDV= option
SVD statement, 55

SVMACHINE procedure, SAVESTATE statement, 50
RSTORE= option, 50

SYNONYM= option
PARSE statement, 49

syntax
BOOLRULE procedure, 11
TEXTMINE procedure, 44
TMSCORE procedure, 88

TARGET statement
TEXTMINE procedure, 56

TARGET= option

DOCINFO statement, 15

TARGETTYPE= option

DOCINFO statement, 15

TERMID= option
PROC BOOLRULE statement, 14

TERMINFO statement
BOOLRULE procedure, 17

TERMINFO= option
PROC BOOLRULE statement, 14

TERMS= option
PROC TMSCORE statement, 89

TERMWGT= option
PARSE statement, 50

TEXTMINE procedure, 44
PARSE statement, 46
PROC TMSCORE statement, 44
SELECT statement, 50
SVD statement, 52

syntax, 44

TEXTMINE procedure, DOC_ID statement, 45

TEXTMINE procedure, PARSE statement, 46
CELLWGT= option, 47
ENTITIES= option, 47
MULTITERM= option, 47
NONG option, 47
NONOUNGROUPS option, 47
NOSTEMMING option, 48
NOTAGGING option, 48
OUTCHILD= option, 48
OUTCONFIG= option, 48
OUTPARENT= option, 48
OUTPOS= option, 48
OUTTERMS= option, 49
REDUCEF= option, 49
SHOWDROPPEDTERMS= option, 49
START= option, 49
STOP= option, 49
SYNONYM= option, 49
TERMWGT= option, 50

TEXTMINE procedure, PROC TEXTMINE statement, 44
DATA= option, 45
LANGUAGE= option, 45
NEWVARNAMES, 45
NTHREADS= option, 45

TEXTMINE procedure, SELECT statement, 50
GROUP= option, 51
IGNORE option, 51
KEEP option, 51
LABELS option, 51

TEXTMINE procedure, SVD statement, 52
COL= option, 53
ENTRY= option, 53
EXACTWEIGHT option, 53
IN_TERMS= option, 53
K= option, 53
KEEPVARS, KEEPVARIABLES, 54
MAX_K= option, 53
NOCUTOFFS option, 53
NUMLABELS= option, 54
OUTDOCPRO= option, 54
OUTTOPICS= option, 54
RES= option, 54
RESOLUTION= option, 54
ROTATION= option, 55
ROW= option, 55
SVDS= option, 55
SVDU= option, 55
SVDV= option, 55
TOL= option, 55

TEXTMINE procedure, TARGET statement, 56

TEXTMINE procedure, VAR statement, 56

TEXTMINE procedure, VARIABLES statement, 56

TMSCORE procedure, 88
PROC TMSCORE statement, 88

syntax, 88

TMSCORE procedure, DOC_ID statement, 90

TMSCORE procedure, PROC TMSCORE statement, 88
DATA= option, 88
DOC= option, 88
OUTPARENT= option, 89
SVDDOCPRO= option, 89
SVDU= option, 89
TERMS= option, 89

TMSCORE procedure, TMSCORE statement
CONFIG= option, 89

TMSCORE procedure, VAR statement, 90

TMSCORE procedure, VARIABLES statement, 90

TOL= option
SVD statement, 55

VAR statement
TEXTMINE procedure, 56
TMSCORE procedure, 90
VARIABLES statement
TEXTMINE procedure, 56
TMSCORE procedure, 90
Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
for additional books and resources.