



THE
POWER
TO KNOW.

**SAS[®] Enterprise Miner[™] and
SAS[®] Text Miner Procedures
Reference for SAS[®] 9.1.3
The DOCPARSE Procedure
(Book Excerpt)**

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., SAS® *Enterprise Miner™* and SAS® *Text Miner Procedures: Reference for SAS® 9.1.3*, Cary, NC: SAS Institute Inc.

SAS® Enterprise Miner™ and SAS® Text Miner Procedures: Reference for SAS 9.1.3

Copyright © 2008 by SAS Institute Inc., Cary, NC, USA.

All rights reserved. Produced in the United States of America.

For a Web download or e-book: Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

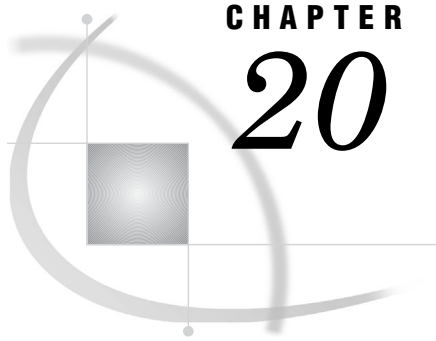
SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, October 2008

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at support.sas.com/publishing or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.



CHAPTER 20

The DOCPARSE Procedure

<i>Overview: DOCPARSE Procedure</i>	411
<i>Syntax: DOCPARSE Procedure</i>	411
<i>PROC DOCPARSE Statement</i>	411
<i>SELECT Statement</i>	418
<i>VAR Statement</i>	419
<i>Examples: DOCPARSE Procedure</i>	420
<i>Example 1: DATA= Option</i>	420
<i>Example 2: OUT= Option</i>	421
<i>Example 3: STEMMING=yes</i>	423
<i>Example 4: NOUNGROUP Option</i>	423
<i>Example 5: NAMEDFILE Option</i>	425
<i>Example 6: OUTOFFSET= Option</i>	425
<i>Example 7: Iterative Parsing and DOC_ID= Option</i>	427
<i>Example 8: SELECT Statement</i>	429

Overview: DOCPARSE Procedure

The DOCPARSE procedure parses text documents in many languages and encodings and organizes the terms and their frequencies into data sets. The DOCPARSE procedure produces two data sets, the KEY data set and the OUT data set. The KEY data set contains a summarized version of the terms within the document collection. Each unique term occurs with a frequency count as well as other information. The OUT data set is a detailed description of every term in each document.

Syntax: DOCPARSE Procedure

```
PROC DOCPARSE <options(s)>;
  SELECT <"pos1"> <"pos2"> ... <"posn"> / <group = "entities" | "attributes" <keep>
    <drop>;
  VAR<textvar>;
```

PROC DOCPARSE Statement

```
PROC DOCPARSE <option(s)>;
```

Required Arguments

KEY=libref.<SAS-data-set>

Specifies one of the SAS data sets that is produced by PROC DOCPARSE. This data set contains a frequency table of the term by document.

The KEY data set contains the following variables:

- TERM — a lowercased version of the term.
- ROLE — the term's part of speech. The ROLE variable is empty if tagging is set to No or if it is not specified in the PROC DOCPARSE statement.
- ATTRIBUTE — for standard terms found in the collection, this variable gives an indication of the characters that compose the term. For entities, the entry merely identifies the term as such. The following is a list of the Attribute types:
 - Alpha — the characters composing the term are all alphabetic characters.
 - Num — at least one character contained in the term is a digit.
 - Punct — The "term" is a punctuation character.
 - Unknown — The term is a mix of alphabetic, punctuation, and white space characters.
 - Entity — The term is an Entity from the entity extraction mechanism.
- FREQ — the number of times the term is found in the document collection.
- NUMDOCS — the number of documents in which the term occurs.
- KEEP — an indicator that identifies whether the term is kept or dropped. The default is Y, and only the stop list can override this default. If the term is in the stop list, KEEP=N.
- KEY — a term number assigned by SAS. Each unique term has a unique key.
- PARENT — the KEY value for the term's parent if the term has a parent; otherwise the value is a period (.). If the values for the KEY, PARENT, and PARENT_ID columns are the same, the observation is the term in its surface form. When the parent occurs as itself, it is referred to as its surface form. The observation is a child term if the values for the PARENT and PARENT_ID columns are the same, but the KEY value is different.
- PARENT_ID — another value that describes the term's parent. The PARENT_ID and the term's KEY values are the same if the PARENT value is a period (.). When this occurs, the observation is either a parent term that represents the summarization of its child terms or the term does not have a parent. If the values in the PARENT_ID and PARENT columns are the same, the term is either in its surface form or it is a child. The term is in its surface form when the PARENT_ID, PARENT, and KEY values are the same. The term is a child when the PARENT_ID and KEY values are different.
- _ISPAR — is a one-character string that indicates how the term is functioning:
 - A plus sign (+) is used to indicate the observation is a parent term.
 - A period (.) is used to indicate that the term is a child.
 - A space () is used to indicate whether the term is functioning independently as neither a parent nor a child.

Note: When stemming or synonym lists are used, child terms are assigned to parent terms with the PARENT and PARENT_ID variables. In addition, many terms are entered on the TERMS table twice. They are entered once in order to indicate how many times the term appeared in the collection as itself and a second time to indicate how many times the term appeared as a

representative for all of its children. The variables PARENT, PARENT_ID, and _ISPAR are all slightly different ways of indicating this information. Δ

OUT=<libref.>SAS-data-set

Specifies one of the SAS data sets that is produced by PROC DOCPARSE. This data set contains information about the terms. The OUT data set contains the following columns:

- _TERMNUM_ — a positive integer assigned by SAS that uniquely identifies a term (or term-role pair when tagging is on).
- _DOCUMENT_ — a positive integer assigned by SAS that serves as a unique key for each document. The documents are assigned the value sequentially beginning with 1.
- _COUNT_ — the number of times the term assigned to _TERMNUM_ was found in the document assigned to _DOCUMENT_.

Options

CONFIG=<libref.>SAS-data-set

Specifies the SAS data set containing configuration information. If this option is specified, the file might or might not exist. If the file does not exist, PROC DOCPARSE creates the file and fills in the columns as specified in the PROC DOCPARSE statement. If the file exists, PROC DOCPARSE uses the options specified in the CONFIG data set. Any PROC DOCPARSE statements take precedence over the options in the CONFIG file. The CONFIG file is updated at the end with the changes stated in PROC DOCPARSE statements. If the CONFIG file exists and you want to change the options, you can edit the CONFIG data set as you would any SAS data set. You can also create a new data set by deleting the CONFIG data set.

Column Name	Format	Description
namedfile	Char 1	Y if the VAR statement specifies a column in an external file that contains the text to parse. N if the VAR statement specifies a column that contains the text to parse.
language	Char 16	Source language of documents. See the LANGUAGE option for a list of supported languages.
encoding	Char 8	Encoding used for the documents (for example, UTF-8 or ASCII).
plugin	Char 12	SAS uses Inxight software by default to parse documents. There is also a simple parser provided by SAS. See the PLUGIN option for more details.

Column Name	Format	Description
stemming	Char 1	If stemming is turned on, the value is Y. If stemming is not turned on, the value is N.
tagging	Char 1	If tagging is turned on, the value is Y. If tagging is not turned on, the value is N.
NG	Char 3	If noun group processing is turned off, the value is OFF. If noun group processing is turned on, the value is set to either MAX, SUB, or STD. See the NOUNGROUP option for details.
entities	Char 1	The value of this field indicates whether entities are turned on or off. It will contain a K if there are entities found in the entList. These entities are kept. The SAS plug-in does not support entities. The Inxight plug-in uses ThingFinder. See the ENTITIES option for details.
entList	Char 200	A string that contains space-delimited entities to keep in parsing. In this version of PROC DOCPARSE, only KEEP is supported.
attributes	Char 1	The value of this field indicates whether attributes found in the attrList are kept or dropped.
attrList	Char 200	A string that contains space-delimited attributes kept or dropped during parsing.
pos	Char 1	The value of this field indicates whether parts of speech found in the posList are kept or dropped.
posList	Char 200	A string that contains space-delimited parts of speech to keep or drop in parsing.

DATA=<libref.>SAS-data-set

Specifies the SAS data set that contains the documents or the file references to parse. If you omit the DATA= option, the procedure uses the value of the `_LAST_ = SAS` system option. The default of `_LAST_ =` is the most recently created SAS data set in the current SAS job or session.

DOC_ID=num

Specifies the starting `_DOCUMENT_` number in the OUT data set. Use this option to perform iterative parsing. See the OUT= option for a detailed description of the contents of the OUT data set.

PROC DOCPARSE supports iterative parsing. To perform iterative parsing, set the INKEY=, NEWKEY= and DOC_ID= options. Suppose you process thousands of observations using PROC DOCPARSE. If you want to add a document to the collection, iterative parsing enables you to add additional documents to a pre-existing KEY data set. You do not need to add the document as an observation to the data set specified in the DATA= option.

ENCODING=NAME8|"QUOTED STRING"

Specifies what encoding the document collection is in. The list of valid encoding values can be found in the <encoding—list> section of the STD.LANGUAGE-ENCODING-CONFIG file. The path to this configuration file is `!sasroot\tmine\sasmisc` where `!sasroot` is the actual path to the SAS 9.1 installation.

Name8	Quoted String	Language Names
WLATIN1	"cp_1252"	English, French, German, Danish, Dutch, Finnish, Italian, Portuguese, Norwegian Bokmal, Spanish, Swedish. Limited results can be obtained on documents in this encoding by setting this encoding and using the English language setting with stemming, tagging, and entities off.
LATIN1	"cp_1252"	English, French, German, Danish, Dutch, Finnish, Italian, Portuguese, Norwegian Bokmal, Spanish, Swedish. Limited results can be obtained on documents in this encoding by setting this encoding and using the English language setting with stemming, tagging, and entities off.
SHIFT-JIS	"shift_jis"	Japanese
BIG5	"big5"	Traditional Chinese
MS-950	"big5"	Traditional Chinese
EUC-CN	"euc_cn"	Simplified Chinese
UTF-8	"utf_8"	Unicode support for each language

Name8	Quoted String	Language Names
EUC-KR	"euc_kr"	Korean
MS-949	"euc_kr"	Korean

Note: The PROC DOCPARSE encoding option must match the encoding option used to start SAS. For example, if you used `-encoding utf-8` to start SAS (for example, `sdssas -dms -autoexec dpauto.sas -encoding utf-8`), use the `encoding='utf_8'` option in PROC DOCPARSE. Δ

ENTITIES=<yes><no>

Specifies whether to find entities. See the SELECT statement for refining the entity search.

IGNORE=<libref>SAS-data-set

Specifies the SAS data set that contains the terms to ignore. An example data set that contains an ignore list can be found in the Sashelp library. It is called STOPLST. Ignore lists and stop lists have the same format.

INKEY=<libref>SAS-data-set

Specifies a SAS data set that was previously generated by PROC DOCPARSE. Use this option to perform iterative parsing. This data set is the starting point when processing the KEY= data set.

LANGUAGE|LANG=NAME16|"QUOTED STRING"

Specifies what language the document collection is in. Use the lowercase version of the language name in quoted strings. This release supports the following languages: Danish, Dutch, English, Finnish, French, German, Italian, Japanese, Korean, Norwegian Bokmal, Portuguese, Simplified Chinese, Spanish, Swedish, and Traditional Chinese.

Use the following quoted strings: "danish", "dutch", "english", "finnish", "french", "german", "italian", "japanese", "korean", "bokmal", "portuguese", "simplified-chinese", "spanish", "swedish", and "traditional-chinese".

MULTITERM=<fileref><quoted string>

Specifies a file reference or a quoted string. The fileref or quoted string specifies the pathname of an XML file that contains a list of multi-word terms. The XML file does not need to have an .xml extension, but it must have the following format:

```
<?xml encoding='cp-1252'?'>
<!-- Sample multi-word token client dictionary -->
<!-- The entries can be deleted if so desired. -->
<multiword-list>

    <item key='WEB BROWSER' />
    <item key='DUE TO' />
    <item key='INCLEMENT WEATHER' />
    <item key='CAPITAL GOODS' />
    <item key='INTEREST RATE' />
</multiword-list>
```

Multi-word terms found in this file are treated as a single term by PROC DOCPARSE. The terms are case-sensitive.

NAMEDFILE

Specifies that the *<textvar>* in the VAR statement is a file reference to the file that contains the text to parse. By default, PROC DOCPARSE assumes that *<textvar>* contains the actual text to parse.

Unlike most of the other options that PROC DOCPARSE writes to the CONFIG = data set, the NAMEDFILE option can change from training time to score time. That is, if at score time the location of the text changes from in the variable to on the file system (or the other way around), you must explicitly change the entry for the NAMEDFILE variable on the data set that is specified in the CONFIG = option.

NOUNGROUPS|NOUNGROUP|NG=<max><sub><std><off>

Specifies whether to find noun groups. The MAX and SUB options are based on Inxight's noun phrases grouping features. The MAX option finds the maximal groups; the SUB option identifies subgroups as well as maximal groups. The STD option is described below. The only modifications to Inxight's grouping features are as follows:

- If stemming is turned on, noun group elements are stemmed. For example, the noun group **amount of defects** is changed to **amount of defect** because the parent of **defects** is **defect**.
- The STD option is the same as SUB except all the noun groups that contain determiners or prepositions (Det or Prep) are omitted.

Default The default is not to report noun groups (NG=OFF).

STEMMING=<yes><no>

Specifies whether stemming is performed. If stemming is turned on, terms might have parents and some terms might also become parents.

TAGGING=<yes><no>

Specifies whether tagging is performed. The tagger identifies the part of speech for each word in a sentence based on its context. In the KEY data set, the Role column contains the tags assigned by PROC DOCPARSE with this option. The Role column is blank when TAGGING=NO. The Attribute column is never omitted.

NEWKEY=<libref.>SAS-data-set

Specifies a SAS data set that contains a frequency table of new terms. Use this option to perform iterative parsing. New terms are terms that are not found in the INKEY data set when processing the DATA data set. The format of the NEWKEY data set is the same as that of the KEY data set. See the KEY= option for a detailed description of the contents of the NEWKEY data set.

NOMULTI

Specifies the parser to break multi-word tokens into their individual units. By default, multi-word tokens are segmented as a single unit. The language-std.multiword file contains a list of multi-word tokens for each language. If a phrase is on this list, it is segmented as one unit. For example, **blood pressure** is segmented as one token. However, by using the NOMULTI option, you can turn this behavior off. In this case, multi-word tokens are broken into their individual units. For example, **blood pressure** is segmented into two distinct units, **blood** and **pressure**, instead of one.

NOPOS

Omits the search for any parts of speech. Use this option when you want to invoke the NOUNGROUP or ENTITY finder only.

OUTOFFSET=<libref.>SAS-data-set

Specifies an output SAS data set that contains offset and length information for a particular term.

PLUG|PLUGIN="QUOTED STRING"

Specifies the parser to use. PLUG="Inxight" or PLUG="SAS" are valid options. SAS provides a simple space-delimited parser.

Default: "Inxight"

RETAIN=<libref.>SAS-data-set

Specifies the SAS data set that contains the terms to retain. Retain lists and ignore lists have the same format.

START=<libref.>SAS-data-set

Specifies the SAS data set that contains the terms to include in your analysis. For items in the START list, the KEEP variable has a value of Y.

STOP=<libref.>SAS-data-set

Specifies the SAS data set that contains the terms to exclude in your analysis. The terms are displayed in the KEY data set. They are not included as parents or rolled up into their respective parent. For items in the STOP list, the KEEP variable has a value of N. See the STOPLST data set in the Sashelp library for an example. Ignore lists and stop lists have the same format.

SYN=<libref.>SAS-data-set

Specifies the SAS data set that contains parent-child relationships. The data set must have the following variables: TERM, PARENT, and CATEGORY. Alternatively, the data set can have the following variables: TERM, PARENT, TERMROLE, and PARENTROLE. This option can be used with or without stemming. If stemming is turned on, then the synonym list overrides any parent-child relationship specified by the natural language processor. If stemming is turned off, the synonym list is used and the parent-child relationships are generated in the KEY data set. No other parent-child relationship, other than the ones specified in the synonym list data set, is generated. An example of a synonym data set named ENGSYNMS can be found in the Sashelp library.

SELECT Statement

```
SELECT <"pos1"> <"pos2"> ... <"posn"> / <group = "entities" | "attributes" <keep>
    <drop>;
```

The SELECT statement specifies what roles and attributes to keep or drop. You must specify either drop or keep. The SELECT statement also specifies what entities to keep. You cannot drop entities.

The *posN* values can be a part of speech or entity category.

Use the following parts-of-speech categories:

- Abbr
- Adj
- Adv
- Aux
- Conj
- Det
- Interj
- Noun
- Num
- Part

- Prep
- Pron
- Prop
- Punct
- Verb
- Verbadj

Use the following entity categories:

- ADDRESS
- COMPANY
- CURRENCY
- DATE
- INTERNET
- LANGUAGE (German and Spanish only)
- LOCATION
- MEASURE
- NOUN_GROUP
- ORGANIZATION
- PEOPLES (German and Spanish only)
- PERCENT
- PERSON
- PHONE
- PRODUCT
- PUBLICATION (German only)
- SSN
- TICKER (English only)
- TIME
- TIME_PERIOD
- TITLE
- VEHICLE

To select entities, use the GROUP= “entities” option. To select attributes, use the GROUP= “attributes” option. To select roles, do not use the GROUP= option.

The GROUP= “entities” option in the SELECT or FILTER clause tells PROC DOCPARSE to pass your list onto the entity extractor. This optimizes search time for entity extraction.

The GROUP= “attributes” option in the SELECT clause tells PROC DOCPARSE to keep or drop terms that match the “attribute” variables.

VAR Statement

VAR <textvar>;

The VAR statement specifies the variable to be parsed. Textvar must be a character variable in the DATA= data set. This statement is required.

Examples: DOCPARSE Procedure

Example 1: DATA= Option

The following example contains three observations:

```
data cars;
  input text $1-70;
  datalines;
I have a white pearl Honda Insight.
Mary has a silver Volkswagen Beetle.
Johnny wants a red Honda Civic Hybrid.
;
run;

proc docparse
  data=work.cars
  entities=yes
  stemming=yes
  tagging=yes
  ng=max
  key=key
  out=out;
var text;
run;

proc sort data=out;
  by _document_;
run;
```

The following example does not use this data set. The DATA= option is implicit. PROC DOCPARSE uses the New_Test data set.

```
data new_test;
  input Text $ 1-70;
  datalines;
I deposited cash and checks in the bank.
I would like to cash a check.
Ducks are parading along the river.
;
run;

proc docparse
  out=myout22
  key=mykey22;
var text;
run;
```

Example 2: OUT= Option

In the following example, PROC DOCPARSE finds all maximal groups, entities, and parts of speech for a single document:

```
data cars;
  input text $1-70;
  datalines;
I have a white pearl Honda Insight.
Mary has a silver Volkswagen Beetle.
Johnny wants a red Honda Civic Hybrid.
;
run;

proc docparse
  data=work.cars
  entities=yes
  stemming=yes
  tagging=yes
  ng=max
  key=dropkey
  out=dropout;
var text;
run;

proc print data=dropkey;
run;

proc print data=dropout;
run;
```

Running this example produces the Work.Dropkey and Work.Dropout data sets. The PROC PRINT output for Work.Dropkey is:

			A		P
			t		a
			t	n	r
			r	u	P e _
			i	m	a n i
T	R		b	F d K	r t s
O e	o		u	r o e	K e _ p
b r	l		t	e c e	e n i a
s m	e		e	q s p	y t d r
1 .	Punct	Punct	3	3 Y 10	. 10
2 a	Det	Alpha	3	3 Y 5	. 5
3 beetle	Prop	Alpha	1	1 Y 17	. 17
4 civic	Prop	Alpha	1	1 Y 22	. 22
5 has	Verb	Alpha	1	1 Y 14	4 4 .
6 have	Verb	Alpha	1	1 Y 4	4 4 .
7 have	Verb	Alpha	2	2 Y 4	. 4 +
8 honda	Prop	Alpha	2	2 Y 8	. 8
9 honda insight	VEHICLE	Entity	1	1 Y 2	. 2
10 hybrid	Prop	Alpha	1	1 Y 23	. 23

```

11 i                      Pron      Alpha  1 1 Y 3 . 3
12 insight                 Prop     Alpha  1 1 Y 9 . 9
13 johnny                  PERSON   Entity 1 1 Y 18 . 18
14 mary                    PERSON   Entity 1 1 Y 11 . 11
15 pearl                   Noun    Alpha  1 1 Y 7 . 7
16 red                     Adj     Alpha  1 1 Y 21 . 21
17 red honda civic hybrid  VEHICLE Entity 1 1 Y 19 . 19
18 silver                   Adj     Alpha  1 1 Y 15 . 15
19 silver volkswagen beetle NOUN_GROUP Unknown 1 1 Y 13 . 13
20 volkswagen              Prop     Alpha  1 1 Y 16 . 16
21 volkswagen beetle       VEHICLE Entity 1 1 Y 12 . 12
22 want                    Verb    Alpha  1 1 Y 24 . 24 +
23 wants                   Verb    Alpha  1 1 Y 20 24 24 .
24 white                   Adj     Alpha  1 1 Y 6 . 6
25 white pearl            NOUN_GROUP Entity 1 1 Y 1 . 1

```

The PROC PRINT output for Work.Dropout is:

Obs	_TERMNUM_	_DOCUMENT_	_COUNT_
1	10	1	1
2	5	1	1
3	4	1	1
4	8	1	1
5	2	1	1
6	3	1	1
7	9	1	1
8	7	1	1
9	6	1	1
10	1	1	1
11	10	2	1
12	5	2	1
13	17	2	1
14	14	2	1
15	11	2	1
16	15	2	1
17	13	2	1
18	16	2	1
19	12	2	1
20	10	3	1
21	5	3	1
22	22	3	1
23	8	3	1
24	23	3	1
25	18	3	1
26	21	3	1
27	19	3	1
28	20	3	1

Example 3: STEMMING=yes

If stemming is turned on, terms might have parents and some terms might also become parents. For example, the term `bottle` is the parent of `bottles`. If the documents include both terms, `bottle` will appear twice in the `KEY` data set: One observation is for `bottle` as a parent term; the other is for `bottle` in its surface form. The `KEY` data set will also include an observation for `bottles`.

The `FREQ` and `NUMDOCS` counts in the `KEY` data set will reflect this. If `bottle` occurs 10 times in 8 separate documents and `bottles` occurs 3 times in 3 separate documents, then `bottle` as a parent occurs a total of 13 times in 11 documents, as follows:

	Term	Role	Attribute	Freq	numdoc	Keep	Key	Parent	Parent_id	_ispar
4	bottle	Noun	Alpha	10	8	Y	3	3	3	+
5	bottle	Noun	Alpha	13	11	Y	3	.	3	+
6	bottles	Noun	Alpha	3	3	Y	17	3	3	

If `bottles` occurs 3 times in 3 documents in which `bottle` also occurs, then `bottle` as a parent occurs a total of 13 times in 8 documents as follows:

	Term	Role	Attribute	Freq	numdoc	Keep	Key	Parent	Parent_id	_ispar
4	bottle	Noun	Alpha	10	8	Y	3	3	3	+
5	bottle	Noun	Alpha	13	8	Y	3	.	3	+
6	bottles	Noun	Alpha	3	3	Y	7	3	3	

The `PARENT`, `PARENT_ID`, and `_ISPAR` columns in the `KEY` data set describe a term's parent information. They are described in the `KEY=` option section.

In the following display, `leak` has four children: `leaks`, `leaking`, `leaked`, and `leak`.

	Term	Role	Attribute	Freq	numdoc	Keep	Key	Parent	Parent_id	_ispar
971	leak	Verb	Alpha	1	1	Y	1517	1517	1517	+
972	leak	Verb	Alpha	14	11	Y	1517	.	1517	+
973	leaked	Verb	Alpha	6	6	Y	217	1517	1517	
974	leaking	Verb	Alpha	5	5	Y	507	1517	1517	
976	leaks	Verb	Alpha	2	2	Y	873	1517	1517	

Example 4: NOUNGROUP Option

Consider the phrase “the blue Honda of Bob.”

If `NG=MAX`, the resulting noun groups are:

- blue honda
- blue honda of bob
- bob

If NG=SUB, the resulting noun groups are:

- blue honda
- blue honda of bob
- bob
- honda of bob

If NG=STD, the resulting noun groups are:

- blue honda
- bob

A good way to evaluate noun groups is to run the following program:

```
data work.cars;
  input text $1-70;
datalines;
the blue Honda of Bob.
;
run;

proc docparse
  data=work.cars nopos
  entities=yes
  noungroup=max
  stemming=yes
  tagging=yes
  key=mykey
  out=myout;
  var text;
run;

proc docparse
  data=work.cars nopos
  entities=yes
  noungroup=sub
  stemming=yes
  tagging=yes
  key=mykey
  out=myout;
  var text;
run;

proc docparse
  data=work.cars nopos
  entities=yes
  noungroup=std
  stemming=yes
  tagging=yes
  key=mykey
  out=myout;
  var text;
run;
```

Example 5: NAMEDFILE Option

If your training data was in a NAMEDFILE and your scoring data was not, you would have to alter the CONFIG file like the following example:

```

/* namedfile location is used on training data*/
proc docparse
data=trainingdata
    config=configdata
    key=mykey
    out=myout
    namedfile;
    var path;
run;

proc sort data=mykey;
    by key;
run;

data merged;
    merge mykey(keep=term role key) pkey;
    by key;
run;

proc sort data=merged force;
    by term role;
run;

proc datasets lib=work nolist;
    modify merged;
    index create both=(term role);
run;

/* change the namedfile attribute from 'Y' to 'N'*/
data configdata;
    set configdata;
    namedfile='N';
run;

/* text is in the scoredata dataset, namedfile is off*/
data _NULL_;
    length text $32767;
    set scoredata;
    rc= docscore(text, "configdata", "merged", "OUT");
run;

```

Example 6: OUTOFFSET= Option

Here is an example that uses the OUTOFFSET= option.

```

data automobiles;
    input text $1-82;
datalines;
I have a white pearl Honda Insight, a white house and many pearls that are white.
Mary has a white pearl Honda Insight.
Johnny wants a a white pearl Honda Insight.
;
run;

proc docparse data=automobiles
    outoffset=zout2
    key=key
    out=out;
var text;
run;

proc sort data=out2;
    by _document_ _offset_;
run;

proc sort data=out;
    by _document_ _termnum_;
run;

```

The contents of the OUTOFFSET data set are shown in the following display:

	TERMNUM	_DOCUMENT_	_COUNT_	_OFFSET_	_LENGTH_
1	8	1	1	34	1
2	3	1	1	7	1
3	2	1	1	2	4
4	6	1	1	21	5
5	1	1	1	0	1
6	7	1	1	27	7
7	5	1	1	15	5
8	4	1	1	9	5
9	8	2	1	35	1
10	3	2	1	9	1
11	13	2	1	29	6
12	10	2	1	5	3
13	9	2	1	0	4
14	11	2	1	11	6
15	12	2	1	18	10
16	8	3	1	37	1
17	3	3	1	13	1
18	17	3	1	25	5
19	6	3	1	19	5
20	18	3	1	31	6
21	14	3	1	0	6
22	16	3	1	15	3
23	15	3	1	7	5

Example 7: Iterative Parsing and DOC_ID= Option

Here is an example of iterative parsing and using the DOC_ID= option::

```

data zdocs;
    input text $1-70;
datalines;
I have a white pearl Honda Insight.
Mary has a silver Volkswagen Beetle.
Johnny wants a red Honda Civic Hybrid.
;
run;

/* adocs contains observations 1 and 2 */
data adocs;
    set zdocs(firstobs=1 obs=2);
run;

/* bdocs contains observation 3 */
data bdocs;
    set zdocs(firstobs=3 obs=3);
run;

/* process all 3 observations as a whole */
proc docparse
    data=zdocs
    key=zkey
    out=zout;
    var text;
run;

/* process the first 2 observations */
proc docparse
    data=adocs
    key=akey
    out=aout;
    var text;
run;

/* now just add one observation to the previously */
/* generated 2 observations */
proc docparse
    inkey=akey
    newkey=newterms
    doc_id=3
    data=bdocs
    key=bkey
    out=bout;
    var text;
run;

/* zkey and bkey should be equal */
proc compare data=zkey compare=bkey;

```

```
run;
```

After running this SAS code, the following data sets are produced. The ZKEY data set contains terms found in documents 1, 2, and 3.

	Term	Role	Attribute	Freq	numdocs	Keep	Key	Parent	Parent_id	_ispar
1	.		Punct	3	3	Y	8	.	8	.
2	a		Alpha	3	3	Y	3	.	3	.
3	beetle		Alpha	1	1	Y	13	.	13	.
4	civic		Alpha	1	1	Y	17	.	17	.
5	has		Alpha	1	1	Y	10	.	10	.
6	have		Alpha	1	1	Y	2	.	2	.
7	honda		Alpha	2	2	Y	6	.	6	.
8	hybrid		Alpha	1	1	Y	18	.	18	.
9	i		Alpha	1	1	Y	1	.	1	.
10	insight		Alpha	1	1	Y	7	.	7	.
11	johnny		Alpha	1	1	Y	14	.	14	.
12	mary		Alpha	1	1	Y	9	.	9	.
13	pearl		Alpha	1	1	Y	5	.	5	.
14	red		Alpha	1	1	Y	16	.	16	.
15	silver		Alpha	1	1	Y	11	.	11	.
16	volkswagen		Alpha	1	1	Y	12	.	12	.
17	wants		Alpha	1	1	Y	15	.	15	.
18	white		Alpha	1	1	Y	4	.	4	.

The AKEY data set contains terms found in documents 1 and 2.

	Term	Role	Attribute	Freq	numdoc	Keep	Key	Parent	Parent_id	_ispar
1	.		Punct	2	2	Y	8	.	8	.
2	a		Alpha	2	2	Y	3	.	3	.
3	beetle		Alpha	1	1	Y	13	.	13	.
4	has		Alpha	1	1	Y	10	.	10	.
5	have		Alpha	1	1	Y	2	.	2	.
6	honda		Alpha	1	1	Y	6	.	6	.
7	i		Alpha	1	1	Y	1	.	1	.
8	insight		Alpha	1	1	Y	7	.	7	.
9	mary		Alpha	1	1	Y	9	.	9	.
10	pearl		Alpha	1	1	Y	5	.	5	.
11	silver		Alpha	1	1	Y	11	.	11	.
12	volkswagen		Alpha	1	1	Y	12	.	12	.
13	white		Alpha	1	1	Y	4	.	4	.

The BKEY data set is identical to the ZKEY data set. The NEWKEY data set contains only terms found in document 3 that were not found in documents 1 and 2.

	Term	Role	Attribute	Freq	numdocs	Keep	Key	Parent	Parent_id	_ispar
1	civic		Alpha	1	1	Y	17	.	17	.
2	hybrid		Alpha	1	1	Y	18	.	18	.
3	johnny		Alpha	1	1	Y	14	.	14	.
4	red		Alpha	1	1	Y	16	.	16	.
5	wants		Alpha	1	1	Y	15	.	15	.

The BOUT_DOCUMENT_ counter begins at 3 and, if appended to AOUT, will be identical to ZOUT.

	TERMNUM	_DOCUMENT_	_COUNT_
1	8	3	1
2	3	3	1
3	17	3	1
4	6	3	1
5	18	3	1
6	14	3	1
7	16	3	1
8	15	3	1

Example 8: SELECT Statement

In the following example, only nouns, verbs, and adjectives are kept. Tagging is turned on by default because the SELECT clause is specified.

```
proc docparse
  data=work.cars
  stemming=yes
  key=mykey
  out=myout;
  var text;
  select "noun" "verb" "adj" / keep;
run;
```

In the following example, only values for vehicle, organization, and company are returned by PROC DOCPARSE. Notice that NOPOS tells PROC DOCPARSE to ignore parts of speech.

```
proc docparse
  data=work.cars
  config=tmp nopos
  entities=yes
  stemming=yes
  tagging=yes
  key=mykey out=myout;
  var text;
  select "VEHICLE" "ORGANIZATION" "COMPANY"/group="entities" keep;
run;

proc sort data=mykey;
  by role;
run;
```