Text Analytics Toolbox™

Examples

MATLAB®

R2017b

MathWorks®

How to Contact MathWorks

Latest news:	www.mathworks.com
Sales and services:	www.mathworks.com/sales_and_services
User community:	www.mathworks.com/matlabcentral
Technical support:	www.mathworks.com/support/contact_us
Phone:	508-647-7000

The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098

Text Analytics Toolbox™ Examples

Trademarks

Patents

Revision History

September 2017	Online Only	New for Version 1.0

Contents

1 Text Analytics Toolbox Examples

1

Text Analytics Toolbox Examples

Create Simple Text Model for Classification

This example shows how to train a simple text classifier on word frequency counts using a bag-of-words model.

You can create a simple classification model which uses word frequency counts as predictors. This example trains a simple classification model to predict whether damage is caused by hail, or thunderstorm wind when given the text data of a weather report.

Load and Extract Text Data

Load the example data. The file `weatherReports.csv` contains weather reports, including a text description and categorical labels for each event.

```
T = readtable("weatherReports.csv",'TextType','string');
```

View the first few rows of the table T.

```
head(T)
```

```
ans=8x16 table
            Time             event_id         state              event_type          da
    ____________________    __________    ________________    ___________________    __

    22-Jul-2016 16:10:00    6.4433e+05    "MISSISSIPPI"       "Thunderstorm Wind"    ""
    15-Jul-2016 17:15:00    6.5182e+05    "SOUTH CAROLINA"    "Heavy Rain"           "2
    15-Jul-2016 17:25:00    6.5183e+05    "SOUTH CAROLINA"    "Thunderstorm Wind"    "0
    16-Jul-2016 12:46:00    6.5183e+05    "NORTH CAROLINA"    "Thunderstorm Wind"    "0
    15-Jul-2016 14:28:00    6.4332e+05    "MISSOURI"          "Hail"                 ""
    15-Jul-2016 16:31:00    6.4332e+05    "ARKANSAS"          "Thunderstorm Wind"    ""
    15-Jul-2016 16:03:00    6.4343e+05    "TENNESSEE"         "Thunderstorm Wind"    "2
    15-Jul-2016 17:27:00    6.4344e+05    "TENNESSEE"         "Hail"                 ""
```

Extract the text data from the field `event_narrative`, and the label data from the field `event_type`.

```
textData = T.event_narrative;
labels = T.event_type;
```

View the first 10 strings in `textData` with the corresponding labels.

```
[string(labels(1:10)) textData(1:10)]
```

```
ans = 10x2 string array
    "Thunderstorm Wind"    "Large tree down between Plante..."
    "Heavy Rain"           "One to two feet of deep standi..."
    "Thunderstorm Wind"    "NWS Columbia relayed a report ..."
    "Thunderstorm Wind"    "Media reported two trees blown..."
    "Hail"                 ""
    "Thunderstorm Wind"    "A few tree limbs greater than ..."
    "Thunderstorm Wind"    "Awning blown off a building on..."
    "Hail"                 "Quarter size hail near Rosemark."
    "Thunderstorm Wind"    "Tin roof ripped off house on O..."
    "Thunderstorm Wind"    "Powerlines down at Walnut Grov..."
```

Extract weather reports that are labelled with "Thunderstorm Wind" or "Hail".

```
idx = labels == "Thunderstorm Wind" | labels == "Hail";
textData = textData(idx);
labels = labels(idx);
```

Prepare Text Data for Analysis

Create a function which tokenizes and preprocesses the text data so it can be used for analysis. The function `preprocessWeatherNarratives`, listed at the end of this example, performs the following steps in order:

1. Erase punctuation.
2. Convert the text data to lowercase.
3. Tokenize the text.
4. Remove a list of stop words.
5. Remove words with 2 or fewer characters.
6. Remove words with 15 or more characters.
7. Normalize the words using the Porter stemmer.

Use `preprocessWeatherNarratives` to prepare the text data.

```
documents = preprocessWeatherNarratives(textData);
```

View the first five preprocessed documents.

```
documents(1:5)
```

```
ans = 
  5x1 tokenizedDocument:
```

```
(1,1)  5 tokens: larg tree down plantersvil nettleton
(2,1)  9 tokens: nw columbia relai report tree blown down tom hall
(3,1) 10 tokens: media report two tree blown down i40 old fort area
(4,1)  0 tokens:
(5,1)  8 tokens: few tree limb greater inch down hwy roseland
```

Create a bag-of-words model from the tokenized documents.

```
bag = bagOfWords(documents)
```

```
bag = 
bagOfWords with 8566 words and 17055 documents:

    larg   tree   down   plantersvil   nettleton   …
       1      1      1             1           1
       0      1      1             0           0
       …
```

Remove words from the bag-of-words model that do not appear more than two times in total.

```
bag = removeInfrequentWords(bag,2)
```

```
bag = 
bagOfWords with 2559 words and 17055 documents:

    larg   tree   down   nw   columbia   …
       1      1      1    0          0
       0      1      1    1          1
       …
```

Remove any documents containing no words from the bag-of-words model, and remove the corresponding entries in `labels`.

```
[bag,idx] = removeEmptyDocuments(bag);
labels(idx) = [];
```

Train Supervised Classifier

Train a supervised classification tree using the word frequency counts from the bag-of-words model and the labels.

Specify the `Counts` property of the bag-of-words model to be the predictors, and `labels` to be the response. To input the word frequency counts into `fitctree`, you must convert the word frequency counts to a full matrix.

```
predictors = full(bag.Counts);
response = labels;
```

Split the data into training and testing partitions using `cvparition`. Specify a holdout test partition of 10%.

```
cvp = cvpartition(response,'HoldOut',0.1);
XTrain = predictors(cvp.training,:);
YTrain = response(cvp.training);
XTest = predictors(cvp.test,:);
YTest = response(cvp.test);
```

Fit a classification tree using the training partition of the data.

```
tree = fitctree(XTrain,YTrain)
```

```
tree = 
  ClassificationTree
             ResponseName: 'Y'
    CategoricalPredictors: []
               ClassNames: {'Hail'  'Thunderstorm Wind'}
           ScoreTransform: 'none'
          NumObservations: 11330


  Properties, Methods
```

Test Classifier

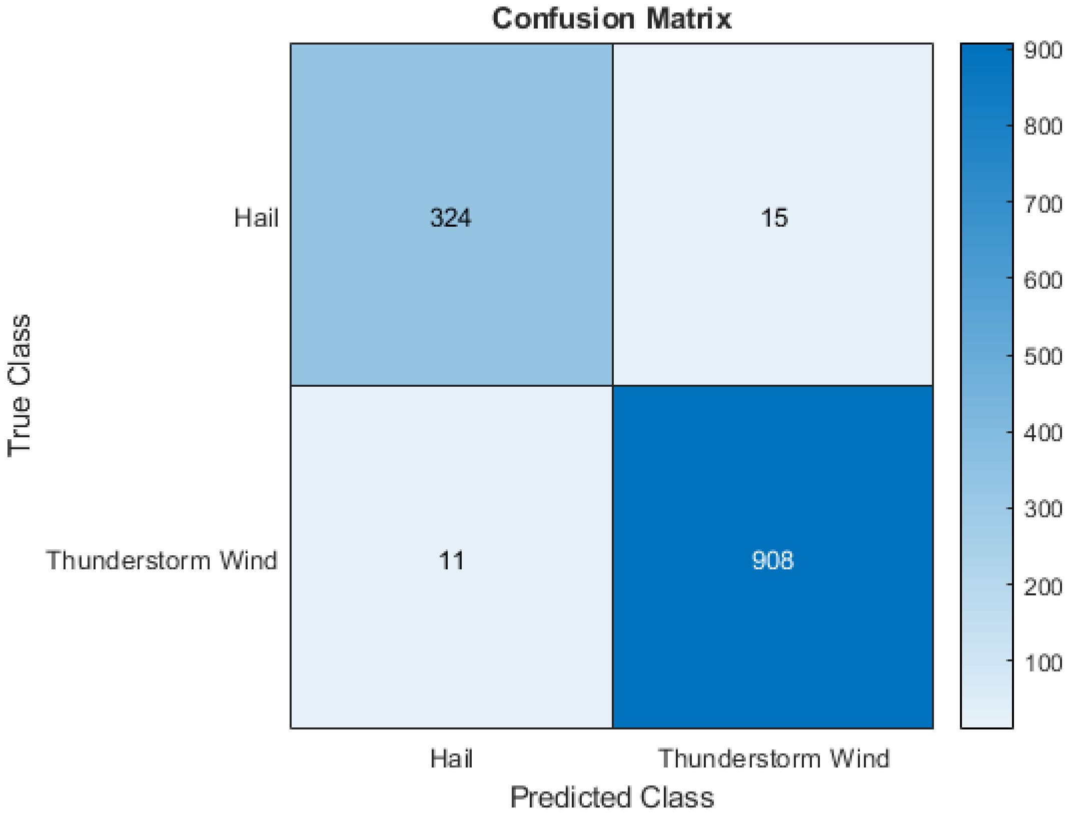
Predict the labels of the test data using the trained classification tree and calculate the classification accuracy. The classification accuracy is the proportion of the labels that the model predicts correctly.

```
YPred = predict(tree,XTest);
acc = sum(YPred == YTest)/numel(YTest)
```

```
acc = 0.9793
```

Create a confusion matrix using `heatmap`.

```
T = table(YPred,YTest);
heatmap(T,"YPred","YTest");
xlabel("Predicted Class")
ylabel("True Class")
title("Confusion Matrix")
```



Example Preprocessing Function

This function performs the following preprocessing steps in order:

1. Erase punctuation.
2. Convert the text data to lowercase.
3. Tokenize the text.

4. Remove a list of stop words.
5. Remove words with 2 or fewer characters.
6. Remove words with 15 or more characters.
7. Normalize the words using the Porter stemmer.

```
function documents = preprocessWeatherNarratives(textData)
% Erase punctuation.
cleanTextData = erasePunctuation(textData);

% Convert the text data to lowercase.
cleanTextData = lower(cleanTextData);

% Tokenize the text.
documents = tokenizedDocument(cleanTextData);

% Remove a list of stop words.
documents = removeWords(documents,stopWords);

% Remove words with 2 or fewer characters, and words with 15 or greater
% characters.
documents = removeShortWords(documents,2);
documents = removeLongWords(documents,15);

% Normalize the words using the Porter stemmer.
documents = normalizeWords(documents);

end
```

See Also

`bagOfWords` | `normalizeWords` | `tokenizedDocument` | `wordcloud`

Related Examples

- “Extract Text Data From Files” on page 1-44
- “Prepare Text Data for Analysis” on page 1-8
- “Visualize Text Data Using Word Clouds” on page 1-18
- “Analyze Text Data Using Topic Models” on page 1-24

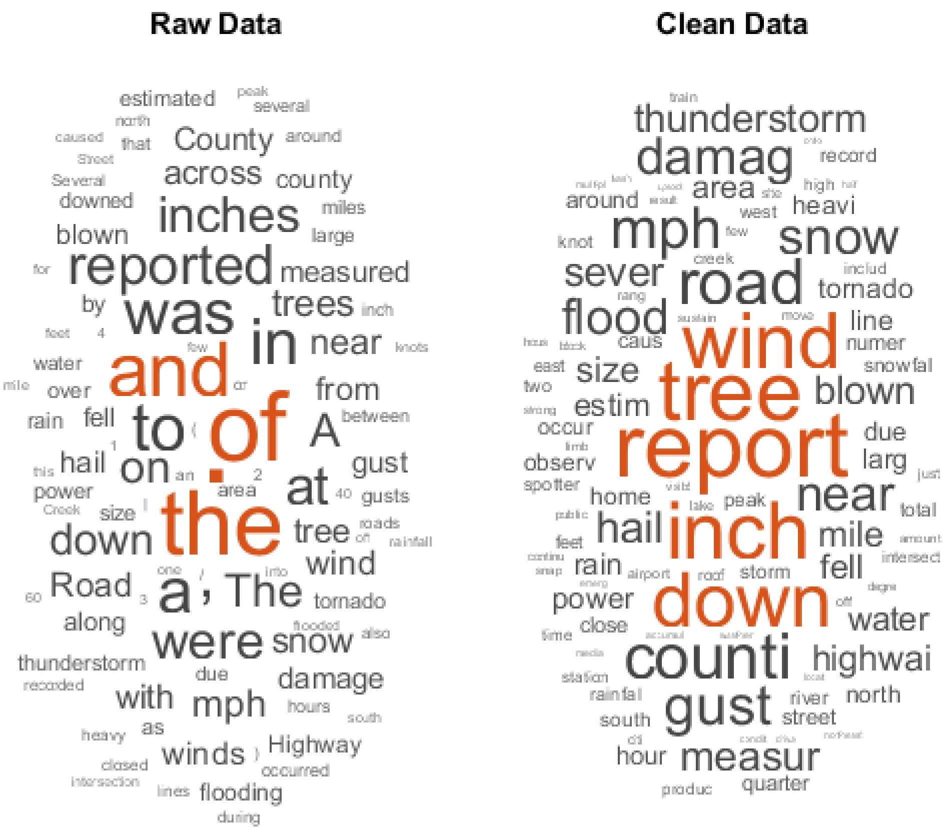
Prepare Text Data for Analysis

This example shows how to create a function which cleans and preprocesses text data for analysis.

Text data can be very large and can contain lots of noise which negatively affects any statistical analysis. For example, text data may contain the following:

- Variations in case, for example "new" and "New"
- Variations in word forms, for example "walk" and "walking"
- Words which add noise, for example stop words such as "the" and "of"
- Punctuation and special characters
- HTML and XML tags

These word clouds illustrate word frequency analysis applied to some raw text data from weather reports, and a preprocessed version of the same text data.



Raw Data

Clean Data

Load and Extract Text Data

Load the example data. The file `weatherReports.csv` contains weather reports, including a text description and categorical labels for each event.

```
T = readtable("weatherReports.csv",'TextType','string');
```

View the first few rows of the table T.

```
head(T)
```

```
ans=8x16 table
            Time                event_id          state                event_type           da
    ________________________    ____________    ___________________    _______________________    __
    22-Jul-2016 16:10:00    6.4433e+05    "MISSISSIPPI"       "Thunderstorm Wind"    ""
    15-Jul-2016 17:15:00    6.5182e+05    "SOUTH CAROLINA"    "Heavy Rain"           "2
    15-Jul-2016 17:25:00    6.5183e+05    "SOUTH CAROLINA"    "Thunderstorm Wind"    "0
    16-Jul-2016 12:46:00    6.5183e+05    "NORTH CAROLINA"    "Thunderstorm Wind"    "0
    15-Jul-2016 14:28:00    6.4332e+05    "MISSOURI"          "Hail"                 ""
    15-Jul-2016 16:31:00    6.4332e+05    "ARKANSAS"          "Thunderstorm Wind"    ""
    15-Jul-2016 16:03:00    6.4343e+05    "TENNESSEE"         "Thunderstorm Wind"    "2
    15-Jul-2016 17:27:00    6.4344e+05    "TENNESSEE"         "Hail"                 ""
```

Extract the text data from the field `event_narrative`, and the label data from the field `event_type`.

```
textData = T.event_narrative;
labels = T.event_type;
```

View the first 10 strings in `textData`.

```
textData(1:10)
```

```
ans = 10x1 string array
    "Large tree down between Plantersville and Nettleton."
    "One to two feet of deep standing water developed on a street on the Winthrop Unive
    "NWS Columbia relayed a report of trees blown down along Tom Hall St."
    "Media reported two trees blown down along I-40 in the Old Fort area."
    ""
    "A few tree limbs greater than 6 inches down on HWY 18 in Roseland."
    "Awning blown off a building on Lamar Avenue. Multiple trees down near the intersec
    "Quarter size hail near Rosemark."
    "Tin roof ripped off house on Old Memphis Road near Billings Drive. Several large t
    "Powerlines down at Walnut Grove and Cherry Lane roads."
```

Prepare String Data for Tokenizing

Erase the punctuation from the text data.

```
cleanTextData = erasePunctuation(textData);
cleanTextData(1:10)
```

```
ans = 10x1 string array
    "Large tree down between Plantersville and Nettleton"
    "One to two feet of deep standing water developed on a street on the Winthrop Unive
    "NWS Columbia relayed a report of trees blown down along Tom Hall St"
    "Media reported two trees blown down along I40 in the Old Fort area"
    ""
    "A few tree limbs greater than 6 inches down on HWY 18 in Roseland"
    "Awning blown off a building on Lamar Avenue Multiple trees down near the intersect
    "Quarter size hail near Rosemark"
    "Tin roof ripped off house on Old Memphis Road near Billings Drive Several large tr
    "Powerlines down at Walnut Grove and Cherry Lane roads"
```

Convert the text data to lowercase.

```
cleanTextData = lower(cleanTextData);
cleanTextData(1:10)
```

```
ans = 10x1 string array
    "large tree down between plantersville and nettleton"
    "one to two feet of deep standing water developed on a street on the winthrop unive
    "nws columbia relayed a report of trees blown down along tom hall st"
    "media reported two trees blown down along i40 in the old fort area"
    ""
    "a few tree limbs greater than 6 inches down on hwy 18 in roseland"
    "awning blown off a building on lamar avenue multiple trees down near the intersect
    "quarter size hail near rosemark"
    "tin roof ripped off house on old memphis road near billings drive several large tr
    "powerlines down at walnut grove and cherry lane roads"
```

Create Tokenized Documents

Create an array of tokenized documents.

```
cleanDocuments = tokenizedDocument(cleanTextData);
cleanDocuments(1:10)
```

```
ans = 
  10x1 tokenizedDocument:

 (1,1)  7 tokens: large tree down between plantersville and nettleton
 (2,1) 37 tokens: one to two feet of deep standing water developed on a street on the w
 (3,1) 13 tokens: nws columbia relayed a report of trees blown down along tom hall st
 (4,1) 13 tokens: media reported two trees blown down along i40 in the old fort area
 (5,1)  0 tokens:
 (6,1) 14 tokens: a few tree limbs greater than 6 inches down on hwy 18 in roseland
 (7,1) 18 tokens: awning blown off a building on lamar avenue multiple trees down near
 (8,1)  5 tokens: quarter size hail near rosemark
 (9,1) 19 tokens: tin roof ripped off house on old memphis road near billings drive sev
(10,1)  9 tokens: powerlines down at walnut grove and cherry lane roads
```

Words like "a", "and", "to", and "the" (known as stop words) can add noise to data. Remove a list of stop words using the `stopWords` and `removeWords` functions.

```
cleanDocuments = removeWords(cleanDocuments,stopWords);
cleanDocuments(1:10)
```

```
ans = 
  10x1 tokenizedDocument:

 (1,1)  5 tokens: large tree down plantersville nettleton
 (2,1) 18 tokens: two feet deep standing water developed street winthrop university cam
 (3,1) 10 tokens: nws columbia relayed report trees blown down tom hall st
 (4,1) 10 tokens: media reported two trees blown down i40 old fort area
 (5,1)  0 tokens:
 (6,1) 10 tokens: few tree limbs greater 6 inches down hwy 18 roseland
 (7,1) 13 tokens: awning blown off building lamar avenue multiple trees down near inter
 (8,1)  5 tokens: quarter size hail near rosemark
 (9,1) 16 tokens: tin roof ripped off house old memphis road near billings drive severa
(10,1)  7 tokens: powerlines down walnut grove cherry lane roads
```

Remove words with 2 or fewer characters, and words with 15 or greater characters.

```
cleanDocuments = removeShortWords(cleanDocuments,2);
cleanDocuments = removeLongWords(cleanDocuments,15);
cleanDocuments(1:10)
```

```
ans = 
  10x1 tokenizedDocument:

 (1,1)  5 tokens: large tree down plantersville nettleton
 (2,1) 18 tokens: two feet deep standing water developed street winthrop university cam
 (3,1)  9 tokens: nws columbia relayed report trees blown down tom hall
 (4,1) 10 tokens: media reported two trees blown down i40 old fort area
 (5,1)  0 tokens:
 (6,1)  8 tokens: few tree limbs greater inches down hwy roseland
 (7,1) 13 tokens: awning blown off building lamar avenue multiple trees down near inter
 (8,1)  5 tokens: quarter size hail near rosemark
 (9,1) 16 tokens: tin roof ripped off house old memphis road near billings drive severa
(10,1)  7 tokens: powerlines down walnut grove cherry lane roads
```

Normalize the words using the Porter stemmer

```
cleanDocuments = normalizeWords(cleanDocuments);
cleanDocuments(1:10)
```

```
ans = 
  10x1 tokenizedDocument:

 (1,1)  5 tokens: larg tree down plantersvil nettleton
 (2,1) 18 tokens: two feet deep stand water develop street winthrop univers campu inch
 (3,1)  9 tokens: nw columbia relai report tree blown down tom hall
 (4,1) 10 tokens: media report two tree blown down i40 old fort area
 (5,1)  0 tokens:
 (6,1)  8 tokens: few tree limb greater inch down hwy roseland
 (7,1) 13 tokens: awn blown off build lamar avenu multipl tree down near intersect wind
 (8,1)  5 tokens: quarter size hail near rosemark
 (9,1) 16 tokens: tin roof rip off hous old memphi road near bill drive sever larg tree
(10,1)  7 tokens: powerlin down walnut grove cherri lane road
```

Create Bag-of-Words Model

Create a bag-of-words model.

```
cleanBag = bagOfWords(cleanDocuments)
```

```
cleanBag = 
bagOfWords with 17816 words and 36176 documents:

    larg   tree   down   plantersvil   nettleton   …
       1      1      1             1           1
       0      0      0             0           0
       …
```

Remove words that do not appear more than two times in the bag-of-words model.

```
cleanBag = removeInfrequentWords(cleanBag,2)
```

```
cleanBag = 
bagOfWords with 6651 words and 36176 documents:

    larg   tree   down   two   feet   …
       1      1      1     0      0
       0      0      0     1      1
       …
```

Some preprocessing steps such as removeInfrequentWords will leave empty documents in the bag-of-words model. To ensure that no empty documents remain in the bag-of-words model after preprocessing, use removeEmptyDocuments as the last step.

Remove empty documents from the bag-of-words model and the corresponding labels from labels.

```
[cleanBag,idx] = removeEmptyDocuments(cleanBag);
labels(idx) = [];
cleanBag
```

```
cleanBag = 
bagOfWords with 6651 words and 28137 documents:

    larg   tree   down   two   feet   …
       1      1      1     0      0
       0      0      0     1      1
       …
```

Create a Preprocessing Function

It can be useful to create a function which performs preprocessing so you can prepare different collections of text data in the same way. For example, you can use a function so that you can preprocess new data using the same steps as the training data.

You can view the function preprocessWeatherNarratives at the end of this example.

Preprocess new text data using preprocessWeatherNarratives.

```
newText = "A tree is downed outside Apple Hill Drive, Natick";
newDocuments = preprocessWeatherNarratives(newText)
```

```
newDocuments = 
  tokenizedDocument:

   7 tokens: tree down outsid appl hill drive natick
```

Compare with Raw Data

Compare the preprocessed data with the raw data.

```
rawDocuments = tokenizedDocument(textData);
rawBag = bagOfWords(rawDocuments)
rawBag = 
bagOfWords with 22742 words and 36176 documents:

    Large   tree   down   between   Plantersville   …
        1      1      1         1               1
        0      0      0         0               0
        …
```

Calculate the reduction in data.

```
cleanNumWords = cleanBag.NumWords
```

```
cleanNumWords = 6651
```

```
rawNumWords = rawBag.NumWords
```

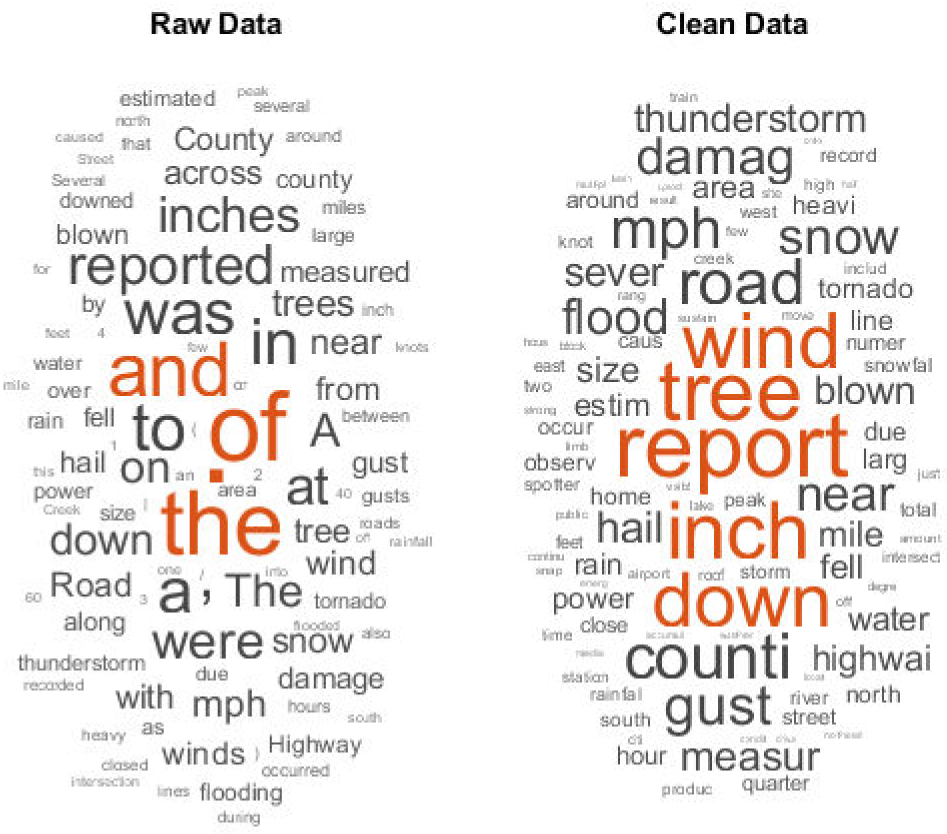
```
rawNumWords = 22742
```

```
reduction = 1 - cleanNumWords/rawNumWords
```

```
reduction = 0.7075
```

Compare the raw data and the cleaned data by visualizing the two bag-of-words models using word clouds.

```
figure
subplot(1,2,1)
wordcloud(rawBag);
title("Raw Data")
subplot(1,2,2)
wordcloud(cleanBag);
title("Clean Data")
```



Example Preprocessing Function

This function performs the preprocessing steps in this example.

```
function [documents] = preprocessWeatherNarratives(textData)
% Erase punctuation.
cleanTextData = erasePunctuation(textData);

% Convert the text data to lowercase.
cleanTextData = lower(cleanTextData);

% Tokenize the text.
documents = tokenizedDocument(cleanTextData);
```

```
% Remove a list of stop words.
documents = removeWords(documents,stopWords);

% Remove words with 2 or fewer characters, and words with 15 or greater
% characters.
documents = removeShortWords(documents,2);
documents = removeLongWords(documents,15);

% Normalize the words using the Porter stemmer.
documents = normalizeWords(documents);
end
```

See Also

`bagOfWords` | `normalizeWords` | `tokenizedDocument` | `wordcloud`

Related Examples

- “Extract Text Data From Files” on page 1-44
- “Create Simple Text Model for Classification” on page 1-2
- “Visualize Text Data Using Word Clouds” on page 1-18
- “Analyze Text Data Using Topic Models” on page 1-24

Visualize Text Data Using Word Clouds

This example shows how to visualize text data using word clouds.

Text Analytics Toolbox extends the functionality of the `wordcloud` (MATLAB) function. It adds support for creating word clouds directly from string arrays, as well as creating word clouds from bag-of-words models and LDA topics.

Load the example data. The file `weatherReports.csv` contains weather reports, including a text description and categorical labels for each event.

```
T = readtable("weatherReports.csv",'TextType','string');
```

View the first few rows of the table T.

```
head(T)
```

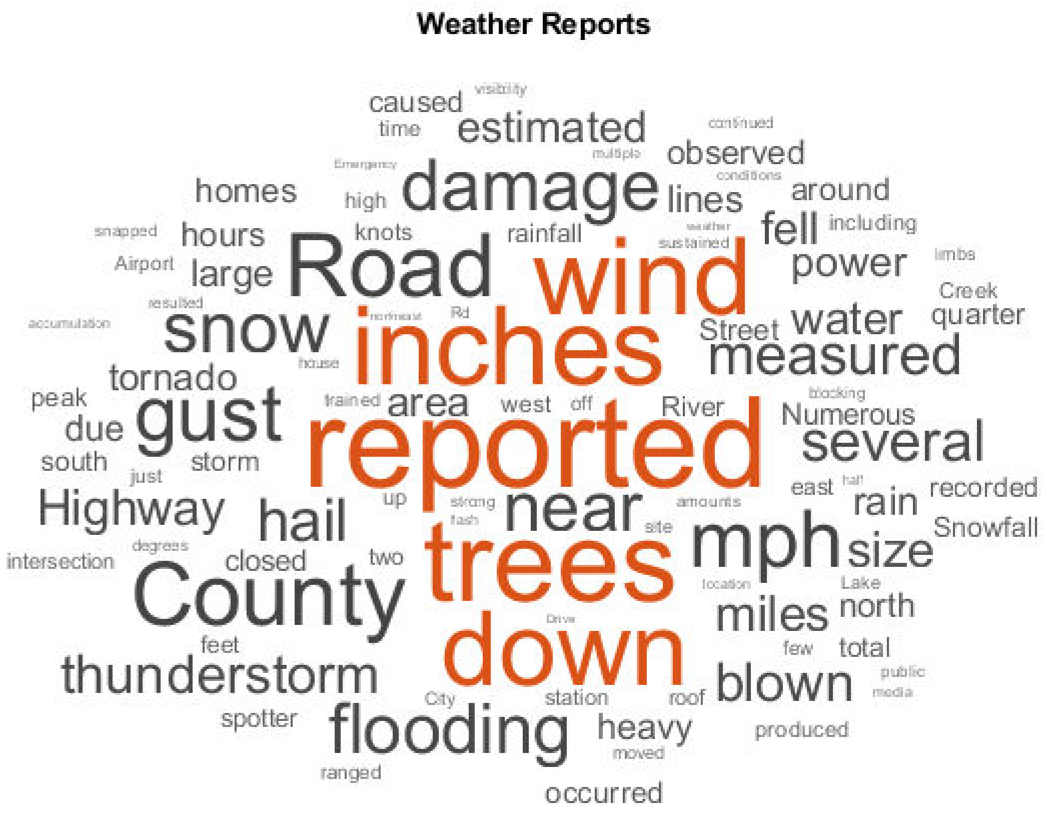
Extract the text data from the `event_narrative` column.

```
textData = T.event_narrative;
textData(1:10)
```

```
ans = 10×1 string array
    "Large tree down between Plantersville and Nettleton."
    "One to two feet of deep standing water developed on a street on the Winthrop Unive
    "NWS Columbia relayed a report of trees blown down along Tom Hall St."
    "Media reported two trees blown down along I-40 in the Old Fort area."
    ""
    "A few tree limbs greater than 6 inches down on HWY 18 in Roseland."
    "Awning blown off a building on Lamar Avenue. Multiple trees down near the intersec
    "Quarter size hail near Rosemark."
    "Tin roof ripped off house on Old Memphis Road near Billings Drive. Several large t
    "Powerlines down at Walnut Grove and Cherry Lane roads."
```

Create a word cloud from all the weather reports.

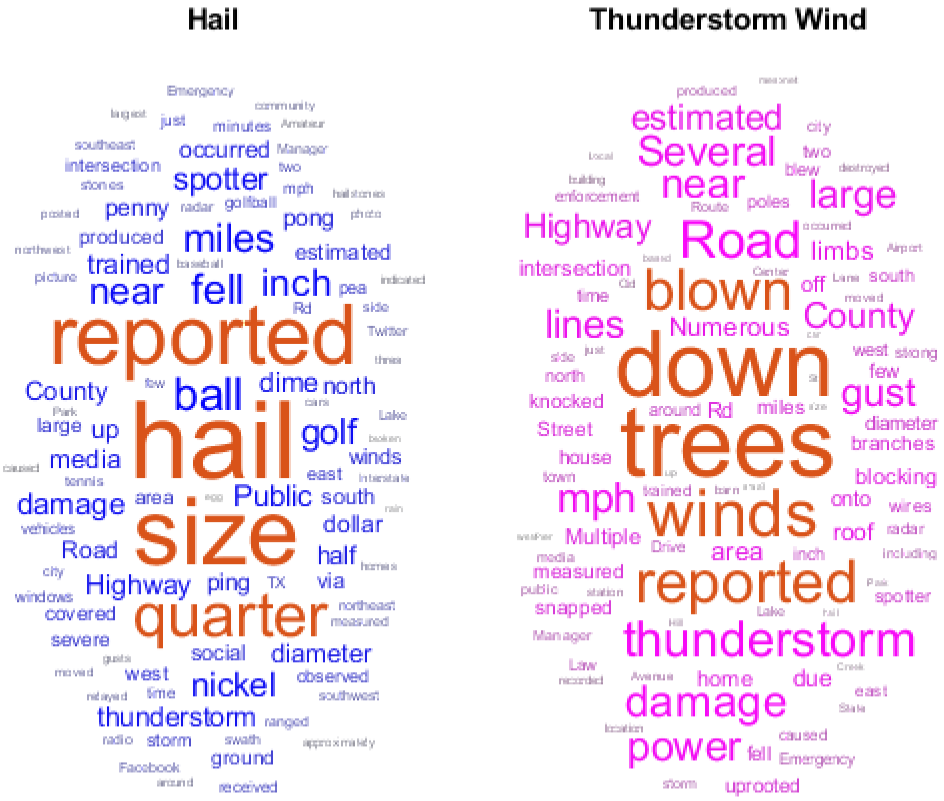
```
figure
wordcloud(textData);
title("Weather Reports")
```



Compare the words in the reports with labels "Hail" and "Thunderstorm Wind". Create word clouds of the reports for each of these labels. Specify the word colors to be blue and magenta for each word cloud respectively.

```
figure
labels = T.event_type;

subplot(1,2,1)
idx = labels == "Hail";
wordcloud(textData(idx),'Color','blue');
title("Hail")

subplot(1,2,2)
idx = labels == "Thunderstorm Wind";
```

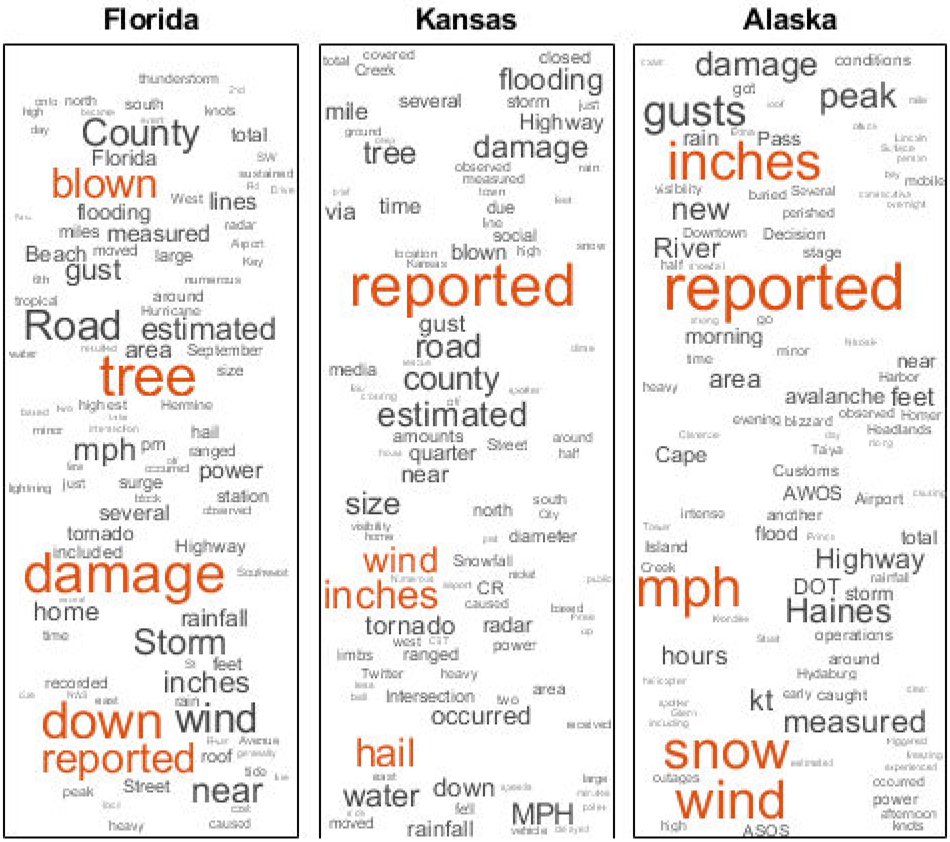
```
wordcloud(textData(idx),'Color','magenta');
title("Thunderstorm Wind")
```



Compare the words in the reports from the states Florida, Kansas, and Alaska. Create word clouds of the reports for each of these states in rectangles and draw a border around each word cloud.

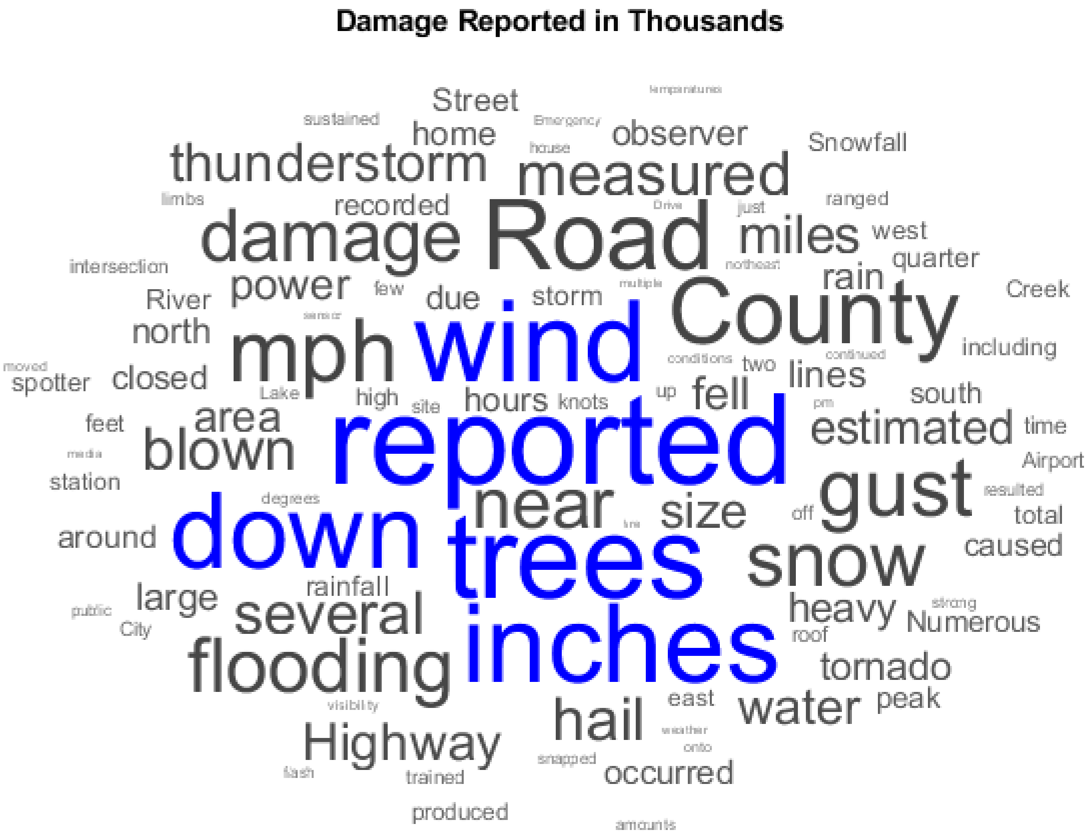
```
figure
state = T.state;

subplot(1,3,1)
idx = state == "FLORIDA";
wordcloud(textData(idx),'Shape','rectangle','Box','on');
title("Florida")
```

```
subplot(1,3,2)
idx = state == "KANSAS";
wordcloud(textData(idx),'Shape','rectangle','Box','on');
title("Kansas")

subplot(1,3,3)
idx = state == "ALASKA";
wordcloud(textData(idx),'Shape','rectangle','Box','on');
title("Alaska")
```

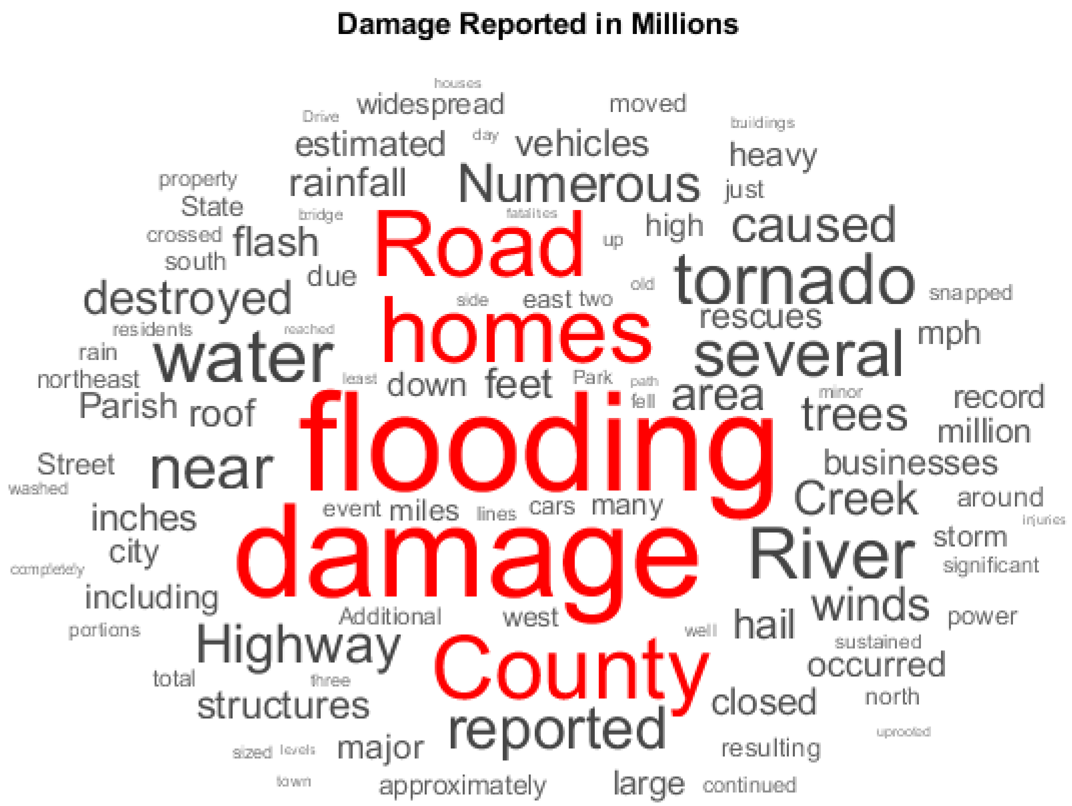


Compare the words in the reports with property damage reported in thousands of dollarsto those with damage reported in millions of dollars. Create word clouds of the reports for each of these amounts with highlight color blue and red respectively.

```
cost = T.damage_property;
idx = endsWith(cost,"K");
figure
wordcloud(textData(idx),'HighlightColor','blue');
title("Damage Reported in Thousands")
```



```
idx = endsWith(cost,"M");
figure
wordcloud(textData(idx),'HighlightColor','red');
title("Damage Reported in Millions")
```


Damage Reported in Millions

See Also

`wordcloud`

Related Examples

- “Extract Text Data From Files” on page 1-44
- “Prepare Text Data for Analysis” on page 1-8
- “Visualize Word Embeddings Using Text Scatter Plots” on page 1-35

Analyze Text Data Using Topic Models

This example shows how to use the Latent Dirichlet Allocation (LDA) topic model to analyze text data.

A Latent Dirichlet Allocation (LDA) model is a topic model which discovers underlying topics in a collection of documents and infers the word probabilities in topics.

To reproduce the results of this example, set `rng` to `'default'`.

```
rng('default')
```

Load and Extract Text Data

Load the example data.

```
load prepTimetable.mat
```

View the first few rows of the timetable `data`.

```
head(data)
```

Extract the text data from the field `event_narrative`.

```
textData = data.event_narrative;
```

View the first 10 strings in `textData`.

```
textData(1:10)
```

```
ans = 10×1 string array
    "Large tree down between Plantersville and Nettleton."
    "One to two feet of deep standing water developed on a street on the Winthrop Unive
    "NWS Columbia relayed a report of trees blown down along Tom Hall St."
    "Media reported two trees blown down along I-40 in the Old Fort area."
    ""
    "A few tree limbs greater than 6 inches down on HWY 18 in Roseland."
    "Awning blown off a building on Lamar Avenue. Multiple trees down near the intersec
    "Quarter size hail near Rosemark."
    "Tin roof ripped off house on Old Memphis Road near Billings Drive. Several large t
    "Powerlines down at Walnut Grove and Cherry Lane roads."
```

Prepare Text Data for Analysis

Create a function which tokenizes and preprocesses the text data so it can be used for analysis. The function `preprocessWeatherNarratives`, listed at the end of this example, performs the following steps in order:

1. Erase punctuation.
2. Convert the text data to lowercase.
3. Tokenize the text.
4. Remove a list of stop words.
5. Remove words with 2 or fewer characters.
6. Remove words with 15 or more characters.
7. Normalize the words using the Porter stemmer.

Use `preprocessWeatherNarratives` to prepare the text data.

```
documents = preprocessWeatherNarratives(textData);
```

View the first 5 preprocessed documents.

```
documents(1:5)
```

```
ans = 
  5×1 tokenizedDocument:

(1,1)  5 tokens: larg tree down plantersvil nettleton
(2,1) 18 tokens: two feet deep stand water develop street winthrop univers campu inch r
(3,1)  9 tokens: nw columbia relai report tree blown down tom hall
(4,1) 10 tokens: media report two tree blown down i40 old fort area
(5,1)  0 tokens:
```

Choose Number of Topics

To decide on a suitable number of topics, you can run `fitlda` for a range of values for the number of topics and compare the perplexity of the each model on a held-out set of documents. The perplexity of an LDA model is a measure of how well the model descibes the data. A lower perplexity suggests a better fit.

Set aside 10% of the documents at random.

```
numDocuments = numel(documents);
cvp = cvpartition(numDocuments,'HoldOut',0.1);
documentsTrain = documents(cvp.training);
documentsTest = documents(cvp.test);
```

Create a bag-of-words model from the training documents. Remove words from the bag-of-words model that have do not appear more than two times in total. Remove any documents containing no words from the bag-of-words model.

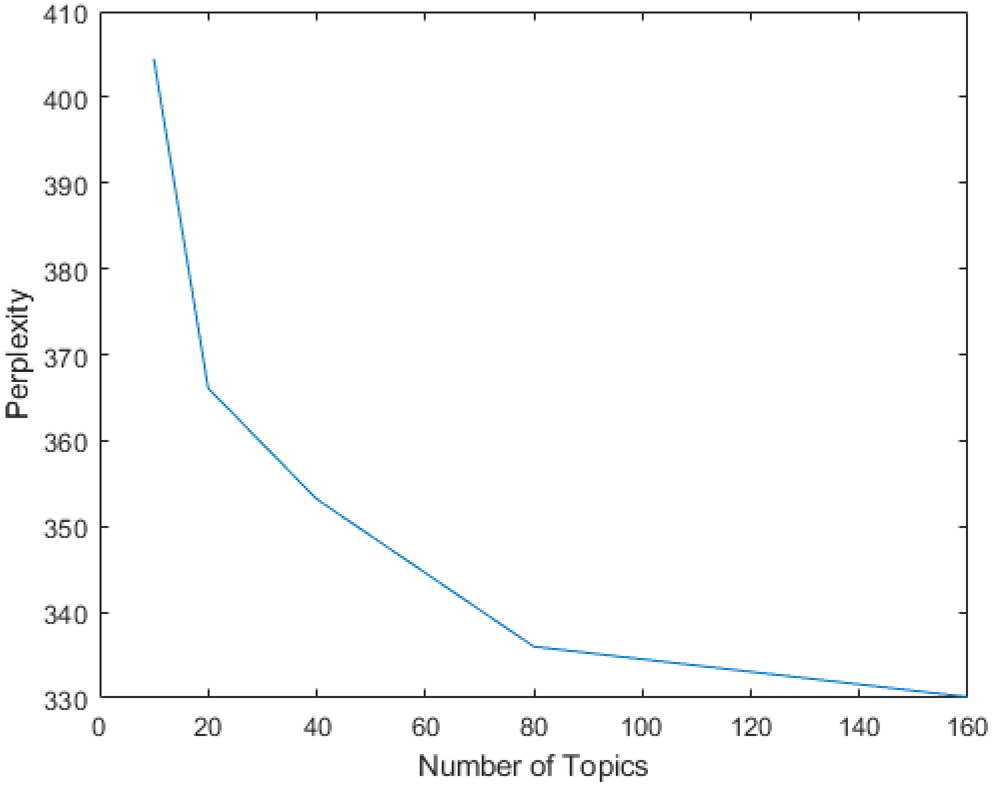
```
bag = bagOfWords(documentsTrain);
bag = removeInfrequentWords(bag,2);
bag = removeEmptyDocuments(bag);
```

Fit LDA models with 10, 20, 40, 80, and 160 topics and calculate the perplexity on the held-out documents. To suppress verbose output, set `'Verbose'` to `0`.

```
numTopicsRange = [10 20 40 80 160];
for i = 1:numel(numTopicsRange)
    numTopics = numTopicsRange(i);
    mdl = fitlda(bag,numTopics, ...
        'Verbose',0);
    [~,ppl(i)] = logp(mdl,documentsTest);
end
```

Show the perplexity for each number of topics in a plot.

```
figure
plot(numTopicsRange,ppl)
xlabel("Number of Topics")
ylabel("Perplexity")
```



The plot suggests that fitting a model with 40 to 80 topics may be a good choice. The perplexity is low compared with the models with fewer topics. Increasing the number of topics may lead to a better fit, but fitting the model may take longer to converge.

Fit LDA Model

Create a bag-of-words model from the tokenized documents.

```
bag = bagOfWords(documents)
```

```
bag = 
bagOfWords with 17816 words and 36176 documents:
    larg   tree   down   plantersvil   nettleton   …
       1      1      1             1           1
       0      0      0             0           0
       …
```

Remove words from the bag-of-words model that have do not appear more than two times in total.

```
bag = removeInfrequentWords(bag,2)
```

```
bag = 
bagOfWords with 6651 words and 36176 documents:

    larg   tree   down   two   feet   …
       1      1      1     0      0
       0      0      0     1      1
       …
```

Remove any documents containing no words from the bag-of-words model.

```
bag = removeEmptyDocuments(bag)
```

```
bag = 
bagOfWords with 6651 words and 28137 documents:

    larg   tree   down   two   feet   …
       1      1      1     0      0
       0      0      0     1      1
       …
```

Fit an LDA model with 60 topics.

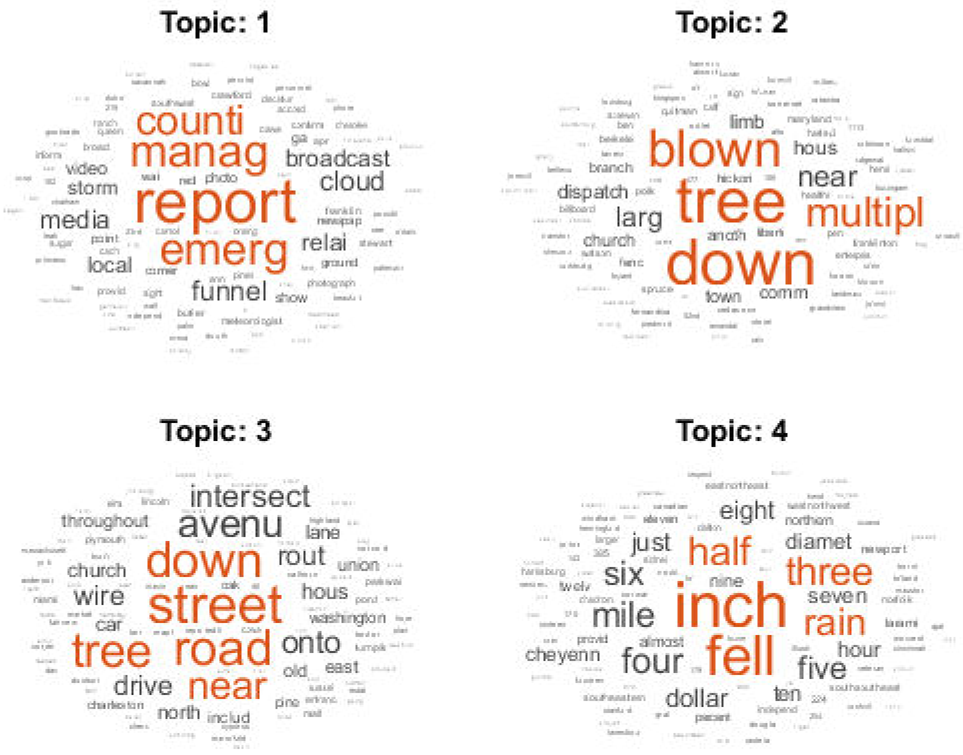
```
numTopics = 60;
mdl = fitlda(bag,numTopics);
```

```
Initial topic assignments sampled in 0.907 seconds.
===============================================================================
| Iteration  |  Time per  | Relative    |  Training  |    Topic    | Concentr.  |
|            |  iter., s  |Delta log(L)| perplexity |  concentr. | iterations |
===============================================================================
|          0 |       0.53 |        Inf |  6.148e+02 |     15.000 |          0 |
|          1 |       3.47 | 1.6625e-01 |  2.461e+02 |     15.000 |          0 |
|          2 |       3.37 | 1.3315e-02 |  2.290e+02 |     15.000 |          0 |
|          3 |       3.65 | 3.6376e-03 |  2.245e+02 |     15.000 |          0 |
|          4 |       3.60 | 2.3478e-03 |  2.217e+02 |     15.000 |          0 |
|          5 |        3.44 | 1.8808e-03 |  2.194e+02 |     15.000 |           0 |
|          6 |        3.51 | 1.4986e-03 |  2.177e+02 |     15.000 |           0 |
|          7 |        3.65 | 1.1233e-03 |  2.164e+02 |     15.000 |           0 |
|          8 |        3.54 | 9.3854e-04 |  2.153e+02 |     15.000 |           0 |
|          9 |        3.40 | 9.3955e-04 |  2.142e+02 |     15.000 |           0 |
|         10 |        3.43 | 1.4198e-03 |  2.126e+02 |     15.000 |           0 |
|         11 |        4.79 | 2.4179e-04 |  2.123e+02 |      6.884 |          18 |
|         12 |        4.37 | 3.8139e-02 |  1.744e+02 |      4.997 |          13 |
|         13 |        4.21 | 1.2765e-02 |  1.634e+02 |      4.402 |          10 |
|         14 |        4.09 | 4.9109e-03 |  1.594e+02 |      4.142 |           8 |
|         15 |        4.28 | 2.7597e-03 |  1.571e+02 |      3.967 |           8 |
|         16 |        4.07 | 1.7692e-03 |  1.557e+02 |      3.864 |           7 |
|         17 |        4.03 | 1.2763e-03 |  1.547e+02 |      3.746 |           7 |
|         18 |        3.98 | 1.6629e-03 |  1.535e+02 |      3.667 |           6 |
|         19 |        3.94 | 1.1975e-03 |  1.525e+02 |      3.578 |           6 |
|         20 |        3.93 | 1.0402e-03 |  1.517e+02 |      3.513 |           6 |
===============================================================================
| Iteration  |  Time per  | Relative   |  Training  |    Topic    | Concentr.  |
|            |  iter., s  |Delta log(L)| perplexity |  concentr. | iterations |
===============================================================================
|         21 |        3.83 | 1.1192e-03 |  1.509e+02 |      3.425 |           6 |
|         22 |        3.83 | 1.4011e-03 |  1.498e+02 |      3.351 |           6 |
|         23 |        4.29 | 1.1501e-03 |  1.490e+02 |      3.287 |           6 |
|         24 |        3.82 | 8.1556e-04 |  1.484e+02 |      3.232 |           5 |
|         25 |        3.93 | 6.4529e-04 |  1.479e+02 |      3.167 |           6 |
|         26 |        3.88 | 1.2284e-03 |  1.470e+02 |      3.122 |           5 |
|         27 |        3.83 | 7.9576e-04 |  1.464e+02 |      3.083 |           5 |
|         28 |        3.73 | 5.7853e-04 |  1.460e+02 |      3.054 |           4 |
|         29 |        3.74 | 4.9358e-04 |  1.456e+02 |      3.007 |           5 |
|         30 |        3.78 | 4.4896e-04 |  1.453e+02 |      2.962 |           5 |
|         31 |        3.83 | 4.5305e-04 |  1.450e+02 |      2.923 |           5 |
|         32 |        3.71 | 6.1762e-04 |  1.445e+02 |      2.884 |           5 |
|         33 |        3.65 | 7.0164e-04 |  1.440e+02 |      2.854 |           4 |
|         34 |        3.71 | 5.0564e-04 |  1.437e+02 |      2.819 |           5 |
|         35 |        3.71 | 2.7122e-04 |  1.435e+02 |      2.795 |           4 |
|         36 |        3.77 | 8.2028e-04 |  1.429e+02 |      2.772 |           4 |
|         37 |        3.74 | 3.0179e-04 |  1.427e+02 |      2.741 |           5 |
|         38 |        3.80 | 4.1905e-04 |  1.424e+02 |      2.711 |           5 |
|         39 |        3.63 | 7.3919e-04 |  1.419e+02 |      2.693 |           4 |
|         40 |        3.78 | 2.3048e-04 |  1.417e+02 |      2.669 |           4 |
===============================================================================
| Iteration  |  Time per  | Relative   |  Training  |    Topic    | Concentr.  |
|            |  iter., s  |Delta log(L)| perplexity |  concentr. | iterations |
===============================================================================
|          41 |        3.77 | 1.1673e-04 |  1.416e+02 |       2.645 |            4 |
|          42 |        3.78 | 9.4803e-05 |  1.415e+02 |       2.614 |            5 |
===============================================================================
```

Vizualize Topics Using Word Clouds

You can use word clouds to easily view the words with the highest probabilities in each topic. Visualize topics 1 through 4 using word clouds.

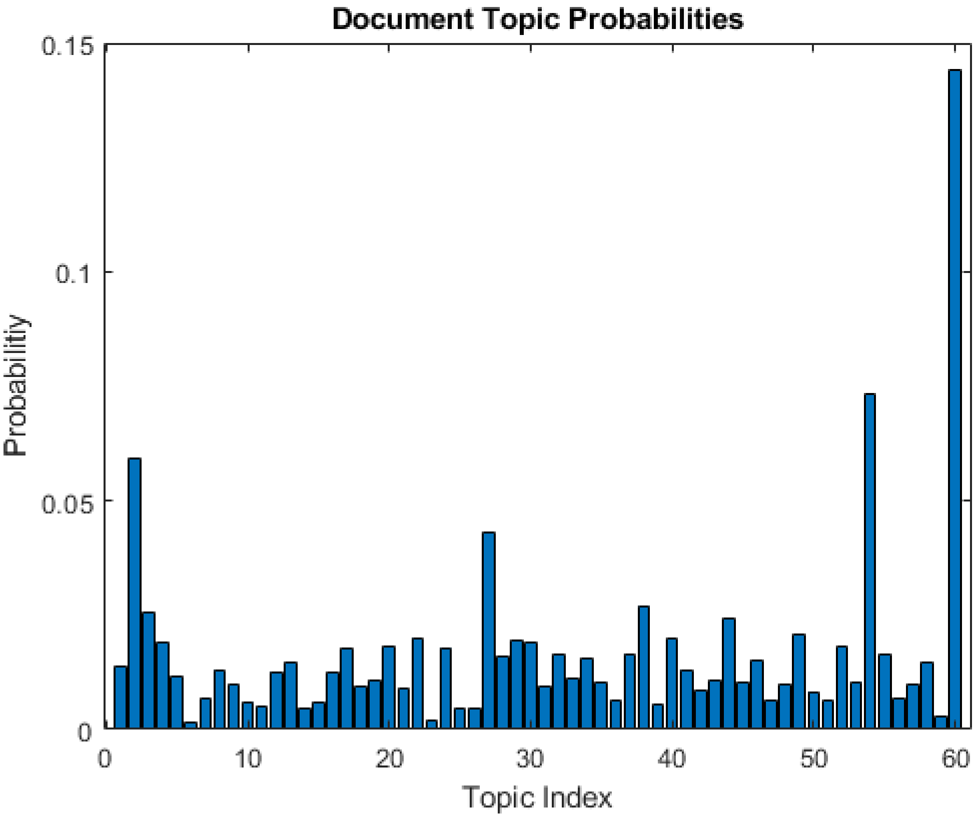
```
figure;
for topicIdx = 1:4
    subplot(2,2,topicIdx)
    wordcloud(mdl,topicIdx);
    title("Topic: " + topicIdx)
end
```



View Mixtures of Topics in Documents

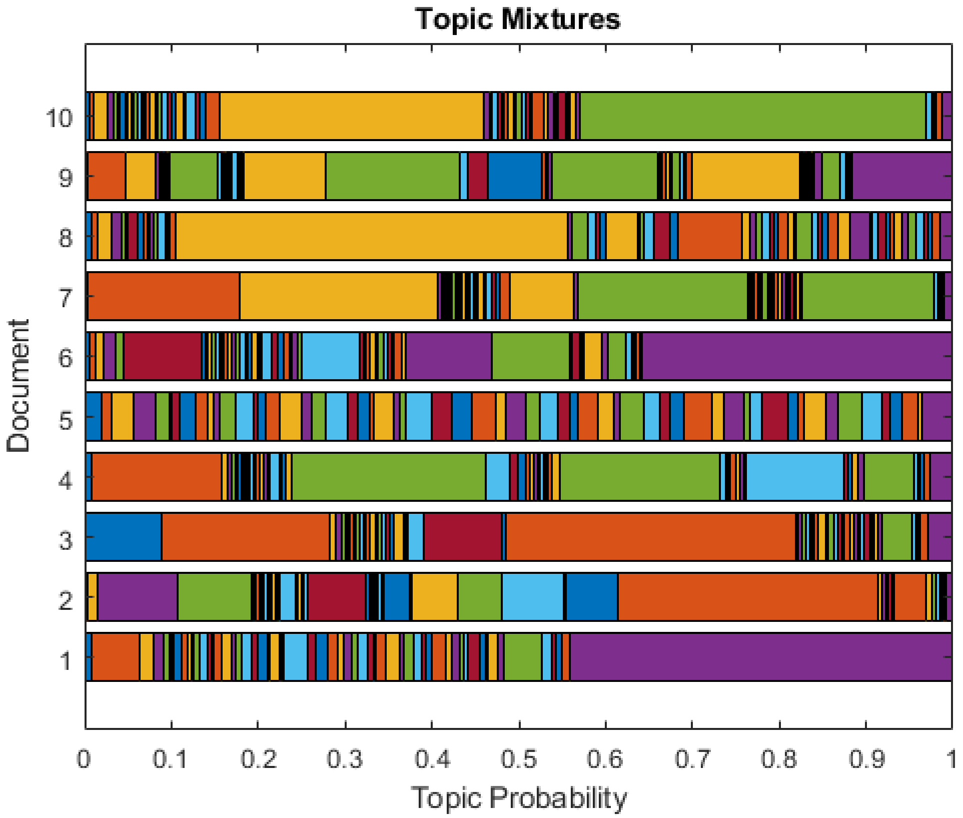
Use `transform` to transform the documents into vectors of topic probabilities.

```
newDocument = tokenizedDocument("A tree is downed outside Apple Hill Drive, Natick");
topicMixture = transform(mdl,newDocument);
figure
bar(topicMixture)
xlabel("Topic Index")
ylabel("Probabilitiy")
title("Document Topic Probabilities")
```



Visualize multiple topic mixtures using stacked bar charts. Visualize the topic mixtures of the first 10 input documents.

```
figure
topicMixtures = transform(mdl,documents(1:10));
barh(topicMixtures(1:10,:),'stacked')
xlim([0 1])
title("Topic Mixtures")
xlabel("Topic Probability")
ylabel("Document")
```



Example Preprocessing Function

This function performs the following preprocessing steps in order:

1. Erase punctuation.
2. Convert the text data to lowercase.
3. Tokenize the text.
4. Remove a list of stop words.
5. Remove words with 2 or fewer characters.
6. Remove words with 15 or more characters.
7. Normalize the words using the Porter stemmer.

```
function [documents] = preprocessWeatherNarratives(textData)
% Erase punctuation.
cleanTextData = erasePunctuation(textData);

% Convert the text data to lowercase.
cleanTextData = lower(cleanTextData);

% Tokenize the text.
documents = tokenizedDocument(cleanTextData);

% Remove a list of stop words.
documents = removeWords(documents,stopWords);

% Remove words with 2 or fewer characters, and words with 15 or greater
% characters.
documents = removeShortWords(documents,2);
documents = removeLongWords(documents,15);

% Normalize the words using the Porter stemmer.
documents = normalizeWords(documents);
end
```

See Also

bagOfWords | fitlda | ldaModel | tokenizedDocument | wordcloud

Related Examples

- “Extract Text Data From Files” on page 1-44
- “Create Simple Text Model for Classification” on page 1-2
- “Prepare Text Data for Analysis” on page 1-8
- “Visualize Text Data Using Word Clouds” on page 1-18

Visualize Word Embeddings Using Text Scatter Plots

This example shows how to visualize word embeddings using 2-D and 3-D t-SNE and text scatter plots.

Word embeddings, map words in a vocabulary to real vectors. The vectors attempt to capture the semantics of the words, so that similar words have similar vectors. Some embeddings also capture relationships between words like "king is to queen as man is to woman". In vector form, this relationship is $king - man + woman = queen$.

To reproduce the results in this example, set `rng` to `'default'`.

```
rng('default')
```

Read the example word embedding. This model was derived by analyzing text from Wikipedia.

```
filename = "exampleWordEmbedding.vec";
emb = readWordEmbedding(filename)
```

```
emb = 
  wordEmbedding with properties:

     Dimension: 50
    Vocabulary: [1×9999 string]
```

Explore the word embedding using `word2vec` and `vec2word`. Convert the words *king*, *man*, and *woman* to vectors using `word2vec`.

```
king = word2vec(emb,"king");
man = word2vec(emb,"man");
woman = word2vec(emb,"woman");
```

Compute the vector given by `king - man + woman`. This vector encapsulates the semantic meaning of the word *king*, without the semantics of the word *man*, and also inludes the semantics of the word *woman*.

```
vec = king - man + woman
```

```
vec = 1×50 single row vector
```

```
   -0.9633    0.2275    0.9614    2.1593   -1.0541   -4.7783   -2.5908   -1.0410   -0.2
```

Find the closest words in the embedding to vec using vec2word.

```
word = vec2word(emb,vec)
```

```
word = 
"queen"
```

Create 2-D Text Scatter Plot

Visualize the word embedding by creating a 2-D text scatter plot using tsne and textscatter.

Convert the words to vectors using word2vec. V is a matrix of word vectors of length 50.

```
words = emb.Vocabulary;
V = word2vec(emb,words);
size(V)
```

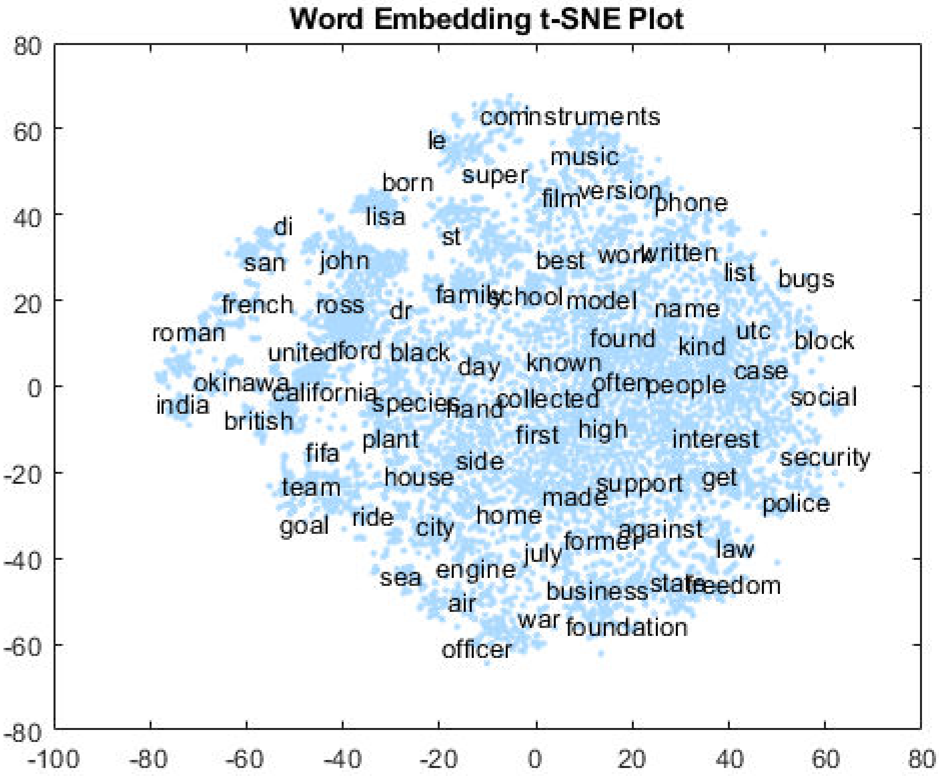
```
ans = 

        9999          50
```

Embed the word vectors in two-dimensional space using tsne. This function may take a few minutes to run. If you want to display the convergence information, then you can set the 'Verbose' name-value pair to 1.

```
XY = tsne(V);
```

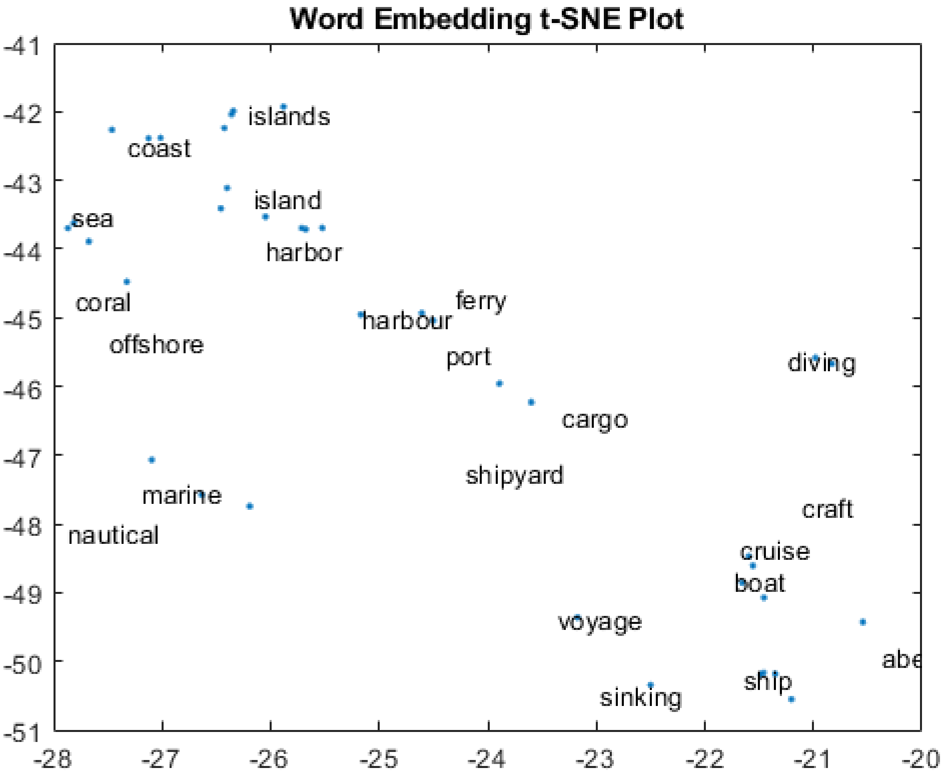
Plot the words at the coordinates specified by XY in a 2-D text scatter plot. For readability, textscatter, by default, does not display all of the input words and displays markers instead.

```
figure
textscatter(XY,words)
title("Word Embedding t-SNE Plot")
```



Zoom in on a section of the plot.

```
xlim([-28 -20])
ylim([-51 -41])
```



Create 3-D Text Scatter Plot

Visualize the word embedding by creating a 3-D text scatter plot using `tsne` and `textscatter`.

Convert the words to vectors using `word2vec`. `V` is a matrix of word vectors of length 50.

```
words = emb.Vocabulary;
V = word2vec(emb,words);
size(V)
```

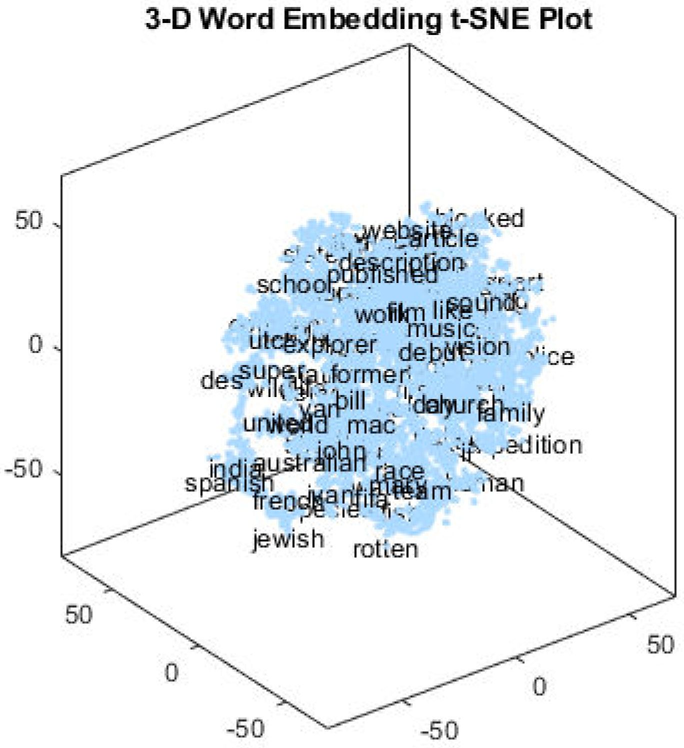
```
ans =
        9999          50
```

Embed the word vectors in a three-dimensional space using `tsne` by specifying the number of dimensions to be three. This function may take a few minutes to run. If you want to display the convergence information, then you can set the `'Verbose'` name-value pair to 1.

```
XYZ = tsne(V, ...
    'NumDimensions',3);
```

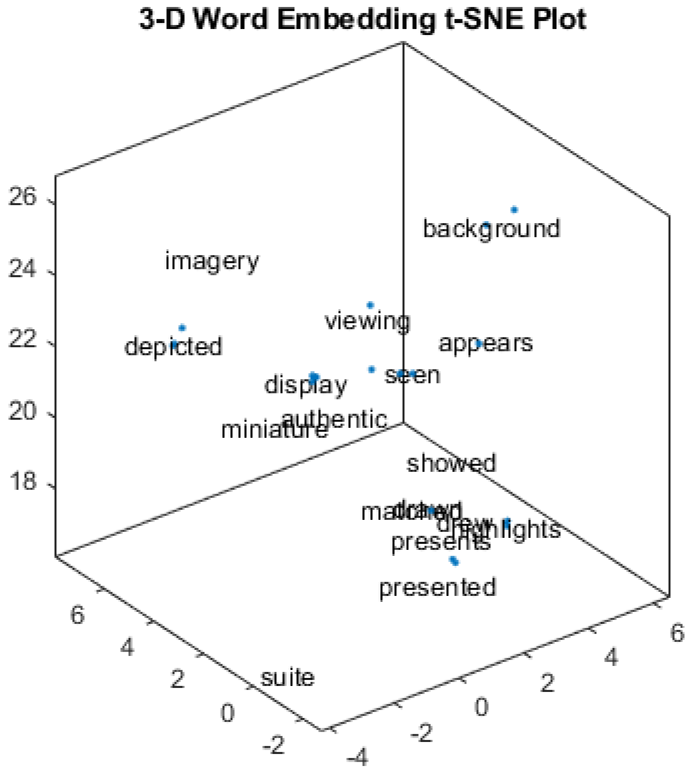
Plot the words at the coordinates specified by XYZ in a 3-D text scatter plot.

```
figure
ts = textscatter3(XYZ,words);
title("3-D Word Embedding t-SNE Plot")
```



Zoom in on a section of the plot.

```
xlim([-4.2 6.5])
ylim([-2.72 7.99])
zlim([16.10 26.81])
```



Perform Cluster Analysis

Convert the words to vectors using `word2vec`. `V` is a matrix of word vectors of length 50.

```
words = emb.Vocabulary;
V = word2vec(emb,words);
size(V)
```

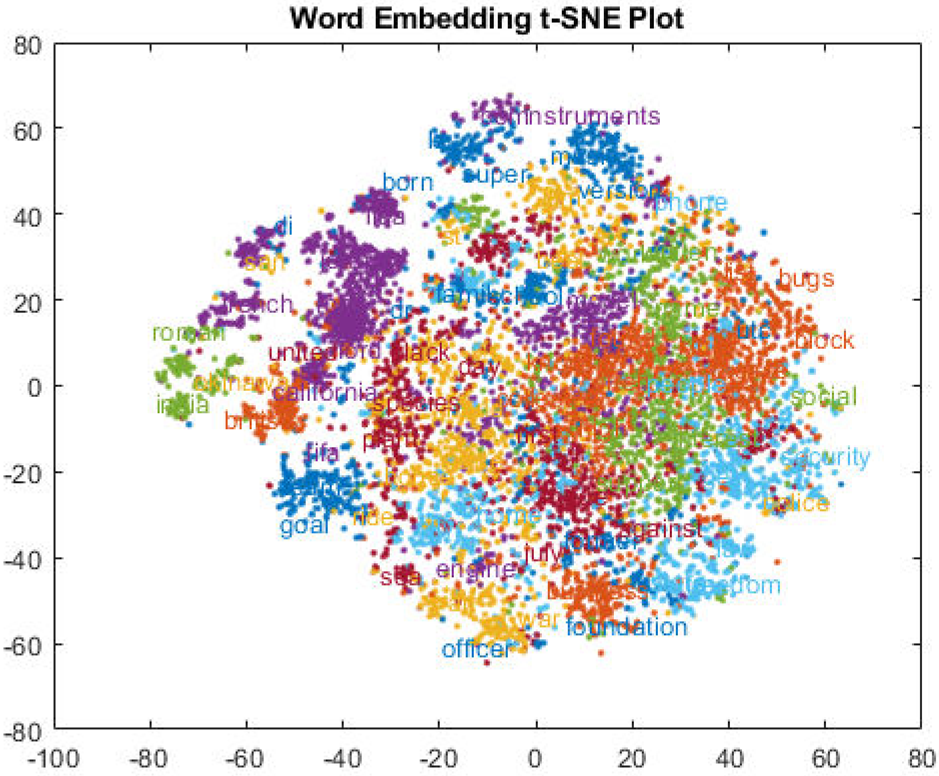
```
ans =

        9999          50
```

Discover 25 clusters using `kmeans`.

```
cidx = kmeans(V,25,'dist','sqeuclidean');
```

Visualize the clusters in a text scatter plot using the 2-D t-SNE data coordinates calculated earlier.

```
figure
textscatter(XY,words, ...
    'ColorData',categorical(cidx));
title("Word Embedding t-SNE Plot")
```



See Also

`readWordEmbedding` | `textscatter` | `textscatter3` | `word2vec` | `wordEmbedding`

Related Examples

- “Extract Text Data From Files” on page 1-44
- “Prepare Text Data for Analysis” on page 1-8
- “Visualize Text Data Using Word Clouds” on page 1-18

Extract Text Data From Files

This example shows how to extract the text data from text, Microsoft Word, PDF, CSV, and Microsoft Excel files and import it into MATLAB for analysis.

Usually, the easiest way to import text data into MATLAB is to use the `extractFileText` function. This function extracts the text data from text, PDF and Microsoft Word files. To import text For CSV and Microsoft Excel files, use `readtable`.

Text Files

Extract the text from `sonnets.txt` using `extractFileText`. The file `sonnets.txt` contains Shakespeare's sonnets in plain text.

```
str = extractFileText("sonnets.txt");
```

View the first sonnet by extracting the text between the two titles "`I`" and "`II`".

```
i = strfind(str,"I");
ii = strfind(str,"II");
start = i(1);
fin = ii(1);
extractBetween(str,start,fin-1)
```

```
ans = 
    "I
    
       From fairest creatures we desire increase,
       That thereby beauty's rose might never die,
       But as the riper should by time decease,
       His tender heir might bear his memory:
       But thou, contracted to thine own bright eyes,
       Feed'st thy light's flame with self-substantial fuel,
       Making a famine where abundance lies,
       Thy self thy foe, to thy sweet self too cruel:
       Thou that art now the world's fresh ornament,
       And only herald to the gaudy spring,
       Within thine own bud buriest thy content,
       And tender churl mak'st waste in niggarding:
         Pity the world, or else this glutton be,
         To eat the world's due, by the grave and thee.
    
       "
```

Microsoft Word Documents

Extract the text from `sonnets.docx` using `extractFileText`. The file `exampleSonnets.docx` contains Shakespeare's sonnets in a Microsoft Word document.

```
str = extractFileText("exampleSonnets.docx");
```

View the second sonnet by extracting the text between the two titles "`II`" and "`III`".

```
ii = strfind(str,"II");
iii = strfind(str,"III");
start = ii(1);
fin = iii(1);
extractBetween(str,start,fin-1)
```

```
ans =
    "II

       When forty winters shall besiege thy brow,

       And dig deep trenches in thy beauty's field,

       Thy youth's proud livery so gazed on now,

       Will be a tatter'd weed of small worth held:

       Then being asked, where all thy beauty lies,

       Where all the treasure of thy lusty days;

       To say, within thine own deep sunken eyes,

       Were an all-eating shame, and thriftless praise.

       How much more praise deserv'd thy beauty's use,

       If thou couldst answer 'This fair child of mine

       Shall sum my count, and make my old excuse,'

       Proving his beauty by succession thine!

         This were to be new made when thou art old,

         And see thy blood warm when thou feel'st it cold.
```

```
        "
```

The example Microsoft Word document uses two newline characters between each line. To replace these with a single newline character, use the `strrep` function.

```
str = strrep(str,[newline newline],newline);
ii = strfind(str,"II");
iii = strfind(str,"III");
start = ii(1);
fin = iii(1);
extractBetween(str,start,fin-1)
```

```
ans = 
    "II
       When forty winters shall besiege thy brow,
       And dig deep trenches in thy beauty's field,
       Thy youth's proud livery so gazed on now,
       Will be a tatter'd weed of small worth held:
       Then being asked, where all thy beauty lies,
       Where all the treasure of thy lusty days;
       To say, within thine own deep sunken eyes,
       Were an all-eating shame, and thriftless praise.
       How much more praise deserv'd thy beauty's use,
       If thou couldst answer 'This fair child of mine
       Shall sum my count, and make my old excuse,'
       Proving his beauty by succession thine!
         This were to be new made when thou art old,
         And see thy blood warm when thou feel'st it cold.
       "
```

PDF Files

Extract the text from `sonnets.docx` using `extractFileText`. The file `exampleSonnets.pdf` contains Shakespeare's sonnets in a PDF.

```
str = extractFileText("exampleSonnets.pdf");
```

View the third sonnet by extracting the text between the two titles "`III`" and "`IV`".

```
iii = strfind(str,"III");
iv = strfind(str,"IV");
start = iii(1);
```

```
fin = iv(1);
extractBetween(str,start,fin-1)
```

```
ans = 
    "III

       Look in thy glass and tell the face thou viewest
       Now is the time that face should form another;
       Whose fresh repair if now thou not renewest,
       Thou dost beguile the world, unbless some mother.
       For where is she so fair whose unear'd womb
       Disdains the tillage of thy husbandry?
       Or who is he so fond will be the tomb,
       Of his self-love to stop posterity?
       Thou art thy mother's glass and she in thee
       Calls back the lovely April of her prime;
       So thou through windows of thine age shalt see,
       Despite of wrinkles this thy golden time.
         But if thou live, remember'd not to be,
         Die single and thine image dies with thee.


       "
```

CSV and Microsoft Excel Files

To extract text data from CSV and Microsoft Excel files, use `readtable` and extract the text data from the table that it returns.

Extract the text from `events_narrative` column of `weatherReports.csv`.

```
T = readtable("weatherReports.csv",'TextType','string');
head(T)
str = T.event_narrative;
str(1:10)
```

```
ans = 10×1 string array
    "Large tree down between Plantersville and Nettleton."
    "One to two feet of deep standing water developed on a street on the Winthrop Unive
    "NWS Columbia relayed a report of trees blown down along Tom Hall St."
    "Media reported two trees blown down along I-40 in the Old Fort area."
    ""
    "A few tree limbs greater than 6 inches down on HWY 18 in Roseland."
    "Awning blown off a building on Lamar Avenue. Multiple trees down near the intersec
    "Quarter size hail near Rosemark."
    "Tin roof ripped off house on Old Memphis Road near Billings Drive. Several large t
    "Powerlines down at Walnut Grove and Cherry Lane roads."
```

Extract Text from Multiple Files

If your text data is contained in multiple files in a directory, then you can import the text data into MATLAB using a file datastore.

Create a file datastore for the three example sonnet text files. The examples sonnets have filenames "`exampleSonnet1.txt`", "`exampleSonnet2.txt`", and "`exampleSonnet3.txt`". Specify the read function to be `extractFileText`.

```
readFcn = @extractFileText
```

```
readFcn = function_handle with value:
    @extractFileText
```

```
fds = fileDatastore('exampleSonnet*.txt', ...
    'ReadFcn',readFcn)
```

```
fds = 
  FileDatastore with properties:

          Files: {
                 ' ...\Documents\MATLAB\examples\textanalytics-ex15735454\exampleSonnet
                 ' ...\Documents\MATLAB\examples\textanalytics-ex15735454\exampleSonnet
                 ' ...\Documents\MATLAB\examples\textanalytics-ex15735454\exampleSonnet
                 }
    UniformRead: 0
        ReadFcn: @extractFileText
```

Loop over the files in the datastore and read each text file.

```
str = [];
while hasdata(fds)
    textData = read(fds);
    str = [str; textData];
end
```

View the extracted text.

```
str
```

```
str = 3×1 string array
    "  From fairest creatures we desire increase,↵  That thereby beauty's rose might n
    "  When forty winters shall besiege thy brow,↵  And dig deep trenches in thy beaut
    "  Look in thy glass and tell the face thou viewest↵  Now is the time that face sh
```

See Also

`extractFileText`

Related Examples

- “Create Simple Text Model for Classification” on page 1-2
- “Prepare Text Data for Analysis” on page 1-8
- “Visualize Text Data Using Word Clouds” on page 1-18
- “Analyze Text Data Using Topic Models” on page 1-24