From Simple Models to Topic Models

- get a big picture of textual data

Outline

- Create Simple Text Model for Classification
  - fitcecoc
- Analyze Text Data Using Multiword Phrases
  - bagOfNgrams
- Analyze Text Data Using Topic Models
  - fitlda

Create Simple Text Model for Classification

- We show how to train a simple text classifier on word frequency counts using a bag-of-words model.
- We create a simple classification model which uses word frequency counts as predictors. This example trains a simple classification model to predict the category of factory reports using text descriptions.

Load and Extract Text Data

- Load the example data. The file factoryReports.csv contains factory reports, including a text description and categorical labels for each report.

```
filename = "factoryReports.csv";
data = readtable(filename,'TextType','string');
head(data)
```

Load and Extract Text Data

head(data)

```
ans =

  8×5 table
```

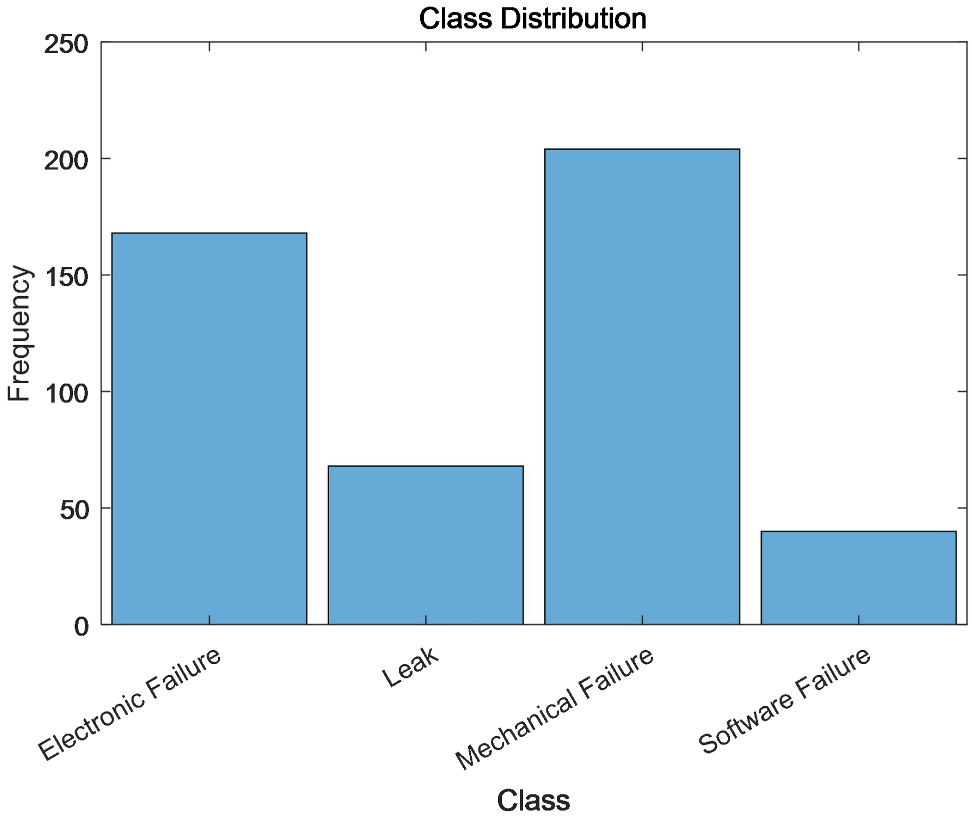
Description	Category	Urgency	Resolution	Cost
"Items are occasionally getting stuck in the scanner spools."	"Mechanical Failure"	"Medium"	"Readjust Machine"	45
"Loud rattling and banging sounds are coming from assembler pistons."	"Mechanical Failure"	"Medium"	"Readjust Machine"	35
"There are cuts to the power when starting the plant."	"Electronic Failure"	"High"	"Full Replacement"	16200
"Fried capacitors in the assembler."	"Electronic Failure"	"High"	"Replace Components"	352
"Mixer tripped the fuses."	"Electronic Failure"	"Low"	"Add to Watch List"	55
"Burst pipe in the constructing agent is spraying coolant."	"Leak"	"High"	"Replace Components"	371
"A fuse is blown in the mixer."	"Electronic Failure"	"Low"	"Replace Components"	441
"Things continue to tumble off of the belt."	"Mechanical Failure"	"Low"	"Readjust Machine"	38

Load and Extract Text Data

- Convert the labels in the Category column of the table to categorical and view the distribution of the classes in the data using a histogram.

```
data.Category = categorical(data.Category);
figure; histogram(data.Category)
xlabel("Class")
ylabel("Frequency")
title("Class Distribution")
```

Load and Extract Text Data



Load and Extract Text Data

- Partition the data into a training partition and a held-out test set. Specify the holdout percentage to be 10%.

```
cvp = cvpartition(data.Category,'Holdout',0.1);
dataTrain = data(cvp.training,:);
dataTest = data(cvp.test,:);
```

Load and Extract Text Data

- Extract the text data and labels from the tables.

```
textDataTrain = dataTrain.Description;
textDataTest = dataTest.Description;
YTrain = dataTrain.Category;
YTest = dataTest.Category;
```

Prepare Text Data for Analysis

- Create a function which tokenizes and preprocesses the text data so it can be used for analysis. The function preprocessText, performs the following steps in order:

1 Tokenize the text using tokenizedDocument.

2 Remove a list of stop words ("and", "of") using removeStopWords.

3 Lemmatize the words using normalizeWords.

4 Erase punctuation using erasePunctuation.

5 Remove words with 2 or fewer characters and with 15 or more characters using removeShortWords and removeLongWords.

Prepare Text Data for Analysis

- Use the example preprocessing function preprocessText to prepare the text data.

```
documents = preprocessText(textDataTrain);
documents(1:5)
```

```
ans =
 5×1 tokenizedDocument:
 6 tokens: items occasionally get stuck scanner spool
 7 tokens: loud rattle bang sound come assembler piston
 4 tokens: cut power start plant
 3 tokens: fry capacitor assembler
 3 tokens: mixer trip fuse
```

Prepare Text Data for Analysis

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

```
bag = bagOfWords(documents)
bag =
 bagOfWords with properties:
 Counts: [432×336 double]
 Vocabulary: [1×336 string]
 NumWords: 336
 NumDocuments: 432
```

Prepare Text Data for Analysis

- Remove words from the bag-of-words model that do not appear more than two times in total. Remove any documents containing no words from the bag-of-words model, and remove the corresponding entries in labels.

```
bag = removeInfrequentWords(bag,2);
[bag,idx] = removeEmptyDocuments(bag);
YTrain(idx) = [];
```

Train Supervised Classifier

- Train a supervised classification model using the word frequency counts from the bag-of-words model and the labels.
- Train a multiclass linear classification model using fitcecoc. Specify the Counts property of the bag-of-words model to be the predictors, and the event type labels to be the response. Specify the learners to be linear. These learners support sparse data input.

```
XTrain = bag.Counts;
mdl = fitcecoc(XTrain,YTrain,'Learners','linear')
```

Error-Correcting Output Codes

Class	Code Word														
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
0	1	1	0	0	0	0	1	0	1	0	0	1	1	0	1
1	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0
2	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1
3	0	0	1	1	0	1	1	1	0	0	0	0	1	0	1
4	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1
5	0	1	0	0	1	1	0	1	1	1	0	0	0	0	1
6	1	0	1	1	1	0	0	0	0	1	0	1	0	0	1
7	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1
8	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1
9	0	1	1	1	0	0	0	0	1	0	1	0	0	1	1

Figure 1: A 15 bit error-correcting output code for a ten-class problem

Test Classifier

- Predict the labels of the test data using the trained model and calculate the classification accuracy. The classification accuracy is the proportion of the labels that the model predicts correctly.
- Preprocess the test data using the same preprocessing steps as the training data. Encode the resulting test documents as a matrix of word frequency counts according to the bag-of-words model.

```
documentsTest = preprocessText(textDataTest);
XTest = encode(bag,documentsTest);
```

Test Classifier

- Predict the labels of the test data using the trained model and calculate the classification accuracy.

```
YPred = predict(mdl,XTest);
acc = sum(YPred == YTest)/numel(YTest)
acc =
   0.8958
```

Predict Using New Data

- Classify the event type of new factory reports.

```
str = [ "Coolant is pooling underneath sorter."
"Sorter blows fuses at start up."
"There are very loud rattling sounds coming from the assembler."];
documentsNew = preprocessText(str);
XNew = encode(bag,documentsNew);
labelsNew = predict(mdl,XNew)
labelsNew = 3×1 categorical
 Leak
 Electronic Failure
 Mechanical Failure
```

Analyze Text Data Using Multiword Phrases

- This example shows how to analyze text using n-gram frequency counts.
- An n-gram is a tuple of n consecutive words. For example, a bigram (the case when n = 2) is a pair of consecutive words such as "heavy rainfall". A unigram (the case when n = 1) is a single word. A bag-of-n-grams model records the number of times that different n-grams appear in document collections.

Analyze Text Data Using Multiword Phrases

- Using a bag-of-n-grams model, you can retain more information on word ordering in the original text data. For example, a bag-of-n-grams model is better suited for capturing short phrases which appear in the text, such as "heavy rainfall" and "thunderstorm winds".
- To create a bag-of-n-grams model, use **bagOfNgrams**. You can input bagOfNgrams objects into other Text Analytics Toolbox functions such as wordcloud and fitlda.

Load and Extract Text Data

- Load the example data. The file factoryReports.csv contains factory reports, including a text description and categorical labels for each event. Remove the rows with empty reports.

```
filename = "factoryReports.csv";
data = readtable(filename,'TextType','String');
```

Load and Extract Text Data

- Extract the text data from the table and view the first few reports.

```
textData = data.Description;
textData(1:5)
ans = 5×1 string
 "Items are occasionally getting stuck in the scanner spools."
 "Loud rattling and banging sounds are coming from assembler pistons."
 "There are cuts to the power when starting the plant."
 "Fried capacitors in the assembler."
 "Mixer tripped the fuses."
```

Load and Extract Text Data

- Use the example preprocessing function preprocessTest to prepare the text data.

```
documents = preprocessText(textData);
documents(1:5)
ans =
 5×1 tokenizedDocument:
 6 tokens: item occasionally get stuck scanner spool
 7 tokens: loud rattle bang sound come assembler piston
 4 tokens: cut power start plant
 3 tokens: fry capacitor assembler ...
```

Create Word Cloud of Bigrams

- Create a word cloud of bigrams by first creating a bag-of-n-grams model using bagOfNgrams.

bag = bagOfNgrams(documents)

bag =

bagOfNgrams with properties:

Counts: [480×941 double]

Vocabulary: [1×351 string]

Ngrams: [941×2 string]

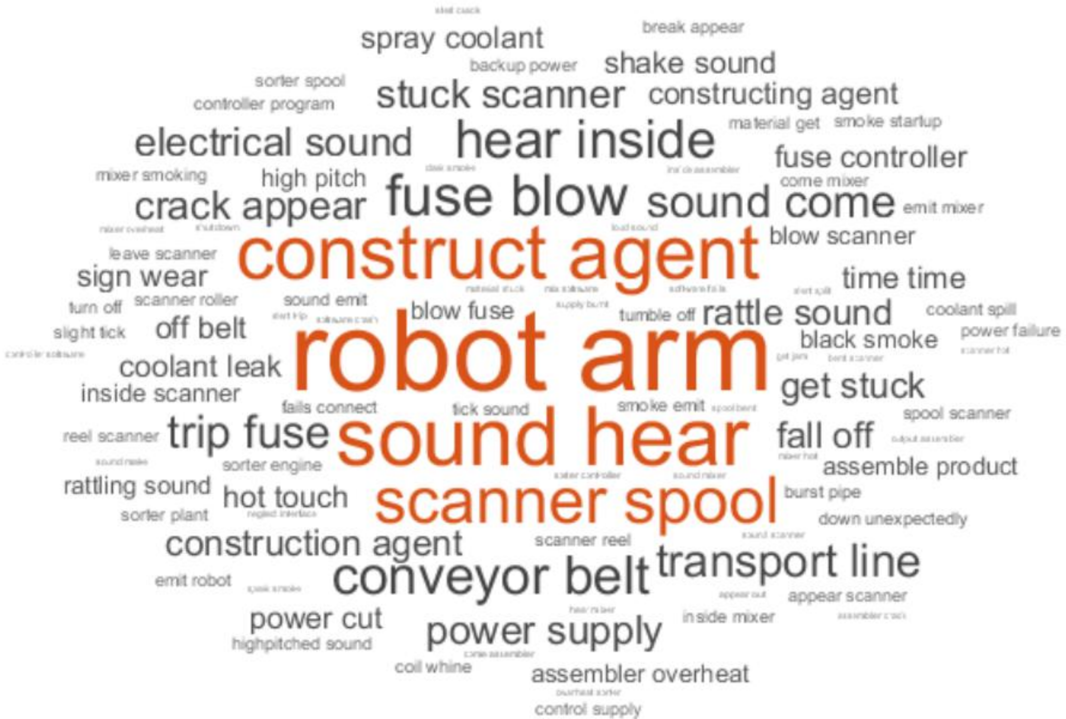
NgramLengths: 2

NumNgrams: 941   NumDocuments: 480

Create Word Cloud of Bigrams

- Visualize the bag-of-n-grams model using a word cloud.

```
figure
wordcloud(bag);
```



Fit Topic Model to Bag-of-N-Grams

- 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.
- Create an LDA topic model with 10 topics using fitlda. The function fits an LDA model by treating the n-grams as single words.

```
mdl = fitlda(bag,10,'Verbose',0);
```

More About Latent Dirichlet Allocation

- A latent Dirichlet allocation (LDA) model is a document topic model which discovers underlying topics in a collection of documents and infers word probabilities in topics. LDA models a collection of D documents as topic mixtures $\theta_1, \ldots, \theta_D$, over K topics characterized by vectors of word probabilities $\phi_1, \ldots, \phi_K$. It assumes that the topic mixtures $\theta_1, \ldots, \theta_D$, and the topics $\phi_1, \ldots, \phi_K$ follow a Dirichlet distribution with concentration parameters α and β respectively.

More About Latent Dirichlet Allocation

- The topic mixtures $\theta_1, \ldots, \theta_D$ are probability vectors of length K, where K is the number of topics. The entry θ_{di} is the probability of topic i appearing in the dth document.
- The topics $\phi_1, \ldots, \phi_K$ are probability vectors of length V, where V is the number of words in the vocabulary. The entry ϕ_{iv} corresponds to the probability of the vth word of the vocabulary appearing in the ith topic.

More About Latent Dirichlet Allocation

- Given the topics $\phi_1, \ldots, \phi_K$ and Dirichlet prior α on the topic mixtures, LDA assumes the following generative process for a document:

1 Sample a topic mixture $\theta \sim \text{Dirichlet}(\alpha)$. The random variable θ is a probability vector of length K, where K is the number of topics.

More About Latent Dirichlet Allocation

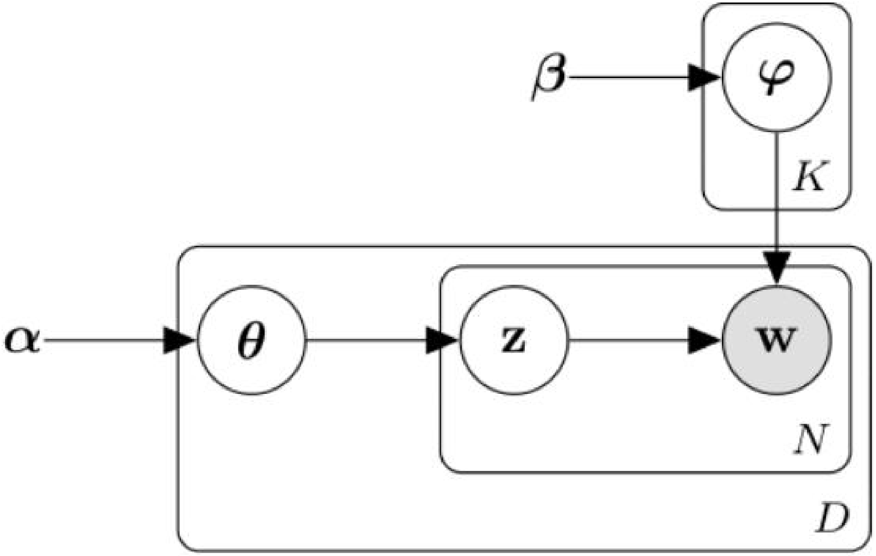
2 For each word in the document:

a) Sample a topic index z ∼ Categorical(θ). The random variable z is an integer from 1 through K, where K is the number of topics.

b) Sample a word w ∼ Categorical(ϕ_z). The random variable w is an integer from 1 through V, where V is the number of words in the vocabulary, and represents the corresponding word in the vocabulary.

More About Latent Dirichlet Allocation

LDA model is a probabilistic graphical model. The shaded nodes are observed variables, unshaded nodes are latent variables, nodes without outlines are model parameters.



Fit Topic Model to Bag-of-N-Grams

- Visualize the first four topics as word clouds. The word clouds highlight commonly co-occurring bigrams in the LDA topics. The function plots the bigrams with sizes according to their probabilities for the specified LDA topics.

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

LDA Topic 1

LDA Topic 2

LDA Topic 3

LDA Topic 4

Analyze Text Using Longer Phrases

- To analyze text using longer phrases, specify the 'NGramLengths' option in bagOfNgrams to be a larger value.
- When working with longer phrases, it can be useful to keep stop words in the model. For example, to detect the phrase "is not happy", keep the stop words "is" and "not" in the model.
- Preprocess the text. Erase the punctuation using erasePunctuation, and tokenize using tokenizedDocument.

```
cleanTextData = erasePunctuation(textData);
documents = tokenizedDocument(cleanTextData);
```

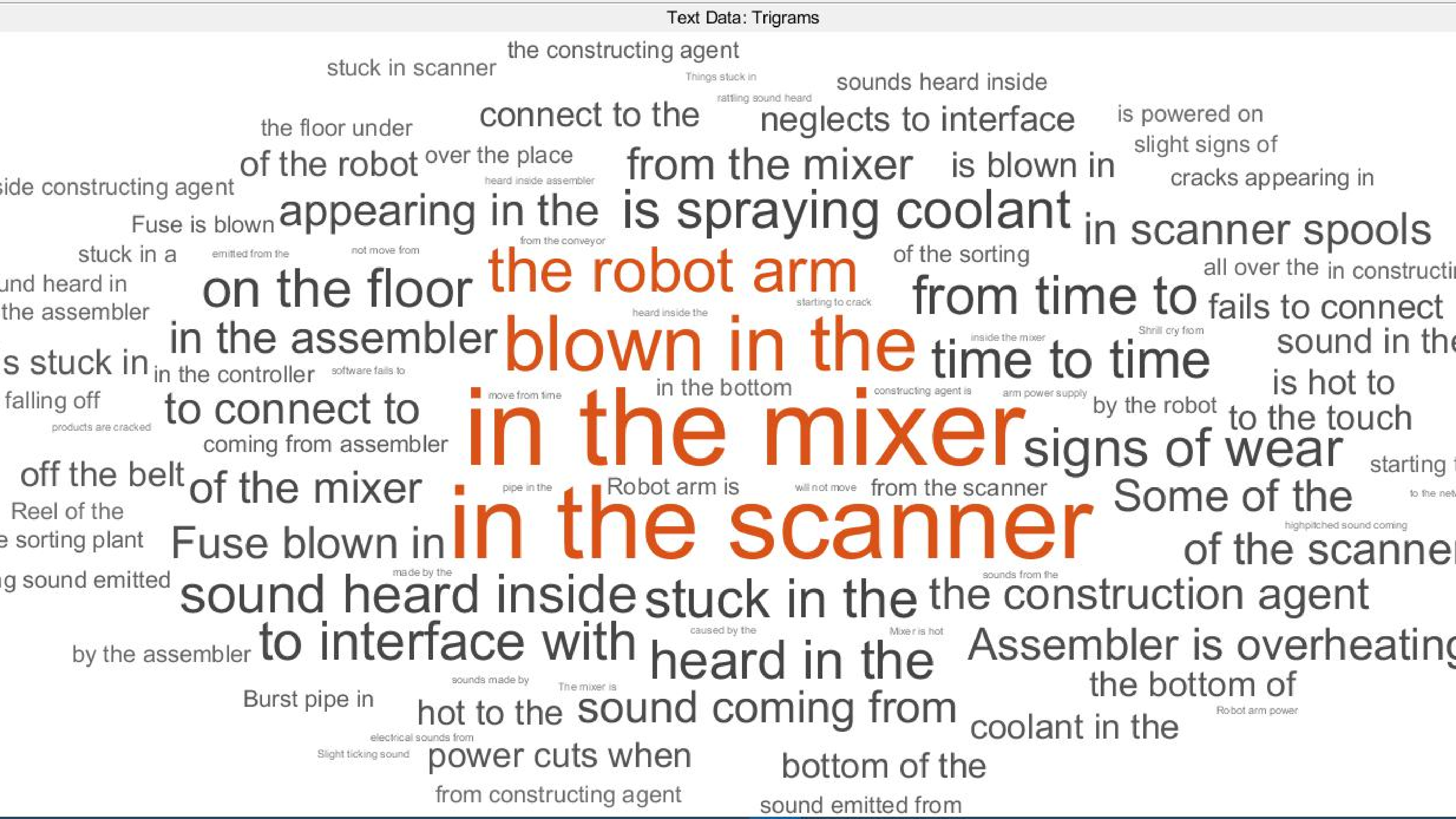
Analyze Text Using Longer Phrases

- To count the n-grams of length 3 (trigrams), use bagOfNgrams and specify 'NGramLengths' to be 3.

```
bag = bagOfNgrams(documents,'NGramLengths',3);
```

- Visualize the bag-of-n-grams model using a word cloud. The word cloud of trigrams better shows the context of the individual words.

```
figure
wordcloud(bag);
title("Text Data: Trigrams")
```



Text Data: Trigrams
in the mixer
in the scanner
blown in the
the robot arm
is spraying coolant
from the mixer
appearing in the
on the floor
in the assembler
from time to
time to time
in scanner spools
signs of wear
neglects to interface
connect to the
of the robot
to connect to
of the mixer
Fuse blown in
sound heard inside
stuck in the
heard in the
to interface with
sound coming from
the construction agent
Assembler is overheating
Some of the
of the scanner
is blown in
fails to connect
the bottom of
coolant in the
bottom of the
power cuts when
hot to the
the constructing agent
stuck in scanner
sounds heard inside
is powered on
slight signs of
cracks appearing in
the floor under
over the place
side constructing agent
Fuse is blown
stuck in a
und heard in
the assembler
s stuck in
in the controller
falling off
off the belt
Reel of the
e sorting plant
g sound emitted
by the assembler
Burst pipe in
coming from assembler
in the bottom
Robot arm is
from the scanner
of the sorting
all over the
in constructi
sound in the
is hot to
by the robot
to the touch
starting t
from constructing agent
sound emitted from
Things stuck in
rattling sound heard
heard inside assembler
from the conveyor
emitted from the
not move from
starting to crack
heard inside the
Shrill cry from
inside the mixer
software fails to
move from time
constructing agent is
arm power supply
products are cracked
pipe in the
will not move
to the netw
highpitched sound coming
made by the
sounds from the
caused by the
Mixer is hot
sounds made by
The mixer is
Robot arm power
electrical sounds from
Slight ticking sound

Analyze Text Using Longer Phrases

- View the top 10 trigrams and their frequency counts using topkngrams.

tbl = topkngrams(bag,10)

tbl=10×3 table

Ngram	Count	NgramLength
"in" "the" "mixer"	14	3
"in" "the" "scanner"	13	3
"blown" "in" "the"	9	3
"the" "robot" "arm"	7	3
"stuck" "in" "the"	6	3
"is" "spraying" "coolant"	6	3
"from" "time" "to"	6	3
"time" "to" "time"	6	3
"heard" "in" "the"	6	3
"on" "the" "floor"	6	3

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.

Load and Extract Text Data

- Load the example data. The file factoryReports.csv contains factory reports, including a text description and categorical labels for each event.

data = readtable("factoryReports.csv",'TextType','string');

head(data)

ans=8×5 table

Description Category Urgency Resolution Cost

"Items are occasionally getting stuck in the scanner spools." "Mechanical Failure" "Medium" "Readjust Machine" 45

"Loud rattling and banging sounds are coming from assembler pistons." "Mechanical Failure" "Medium" "Readjust Machine" 35 ...

-

Load and Extract Text Data

- Extract the text data from the field Description.

```
textData = data.Description;
textData(1:5)
ans = 5×1 string
 "Items are occasionally getting stuck in the scanner spools."
 "Loud rattling and banging sounds are coming from assembler pistons."
 "There are cuts to the power when starting the plant."
 "Fried capacitors in the assembler."
 "Mixer tripped the fuses."
```

Prepare Text Data for Analysis

- Use the preprocessing function preprocessText to prepare the text data.

```
documents = preprocessText(textData);
documents(1:5)
ans =
 5×1 tokenizedDocument:
 6 tokens: items occasionally get stuck scanner spool
 7 tokens: loud rattle bang sound come assembler piston
 4 tokens: cut power start plant
 3 tokens: fry capacitor assembler
 3 tokens: mixer trip fuse
```

Prepare Text Data for Analysis

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

bag = bagOfWords(documents)

bag =

bagOfWords with properties:

Counts: [480×351 double]

Vocabulary: [1×351 string]

NumWords: 351

NumDocuments: 480

Prepare Text Data for Analysis

- 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 = removeInfrequentWords(bag,2);
bag = removeEmptyDocuments(bag)
bag =
 bagOfWords with properties:
 Counts: [480×162 double]
 Vocabulary: [1×162 string]
 NumWords: 162
 NumDocuments: 480
```

Fit LDA Model

- Fit an LDA model with 7 topics.

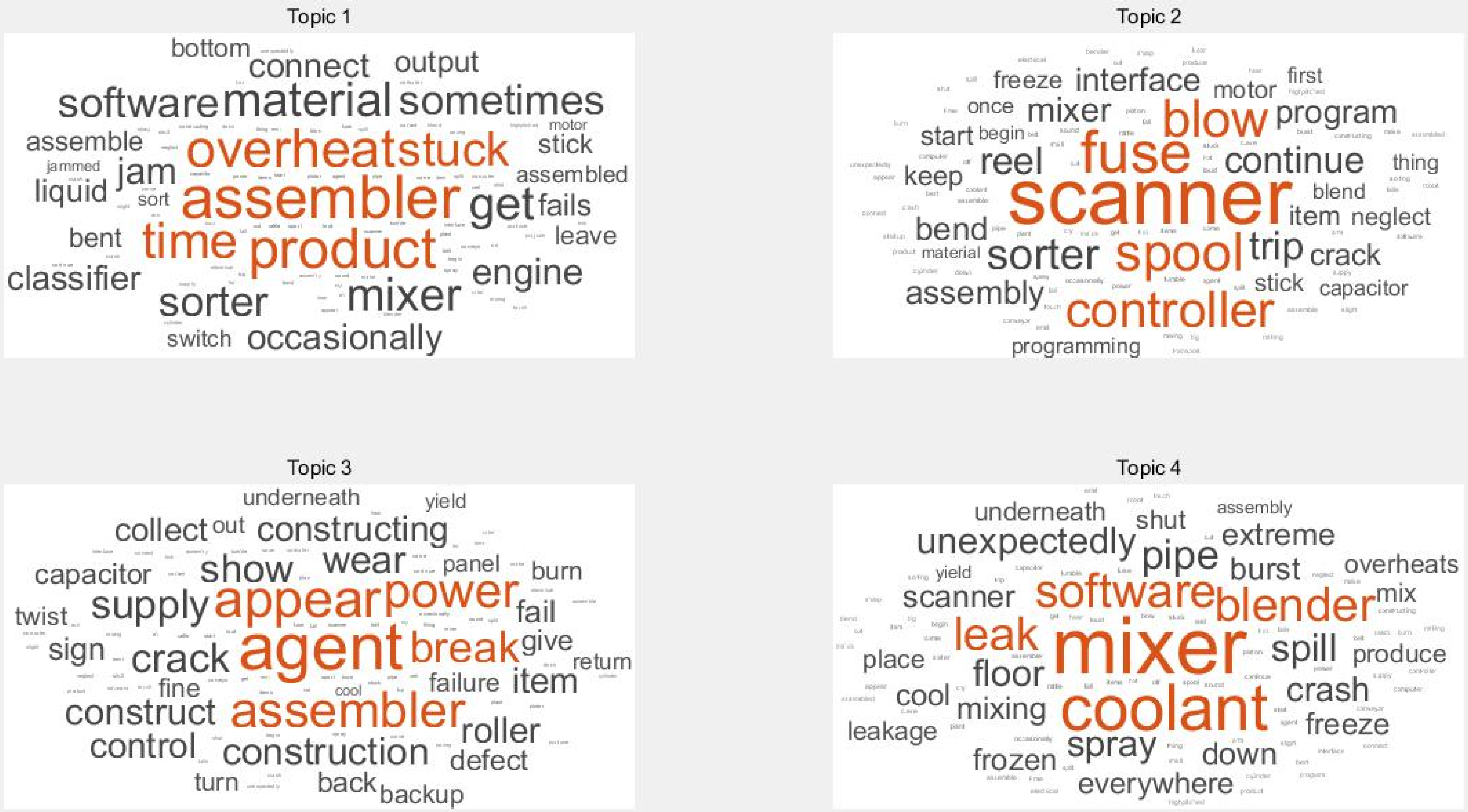
```
numTopics = 7;
mdl = fitlda(bag,numTopics,'Verbose',0);
```

- If you have a large dataset, then the stochastic approximate variational Bayes solver is usually better suited as it can fit a good model in fewer passes of the data. The default solver for fitlda (collapsed Gibbs sampling) can be more accurate at the cost of taking longer to run. To use stochastic approximate variational Bayes, set the 'Solver' option to 'savb'.

Visualize Topics Using Word Clouds

- You can use word clouds to view the words with the highest probabilities in each topic. Visualize the first four topics using word clouds.

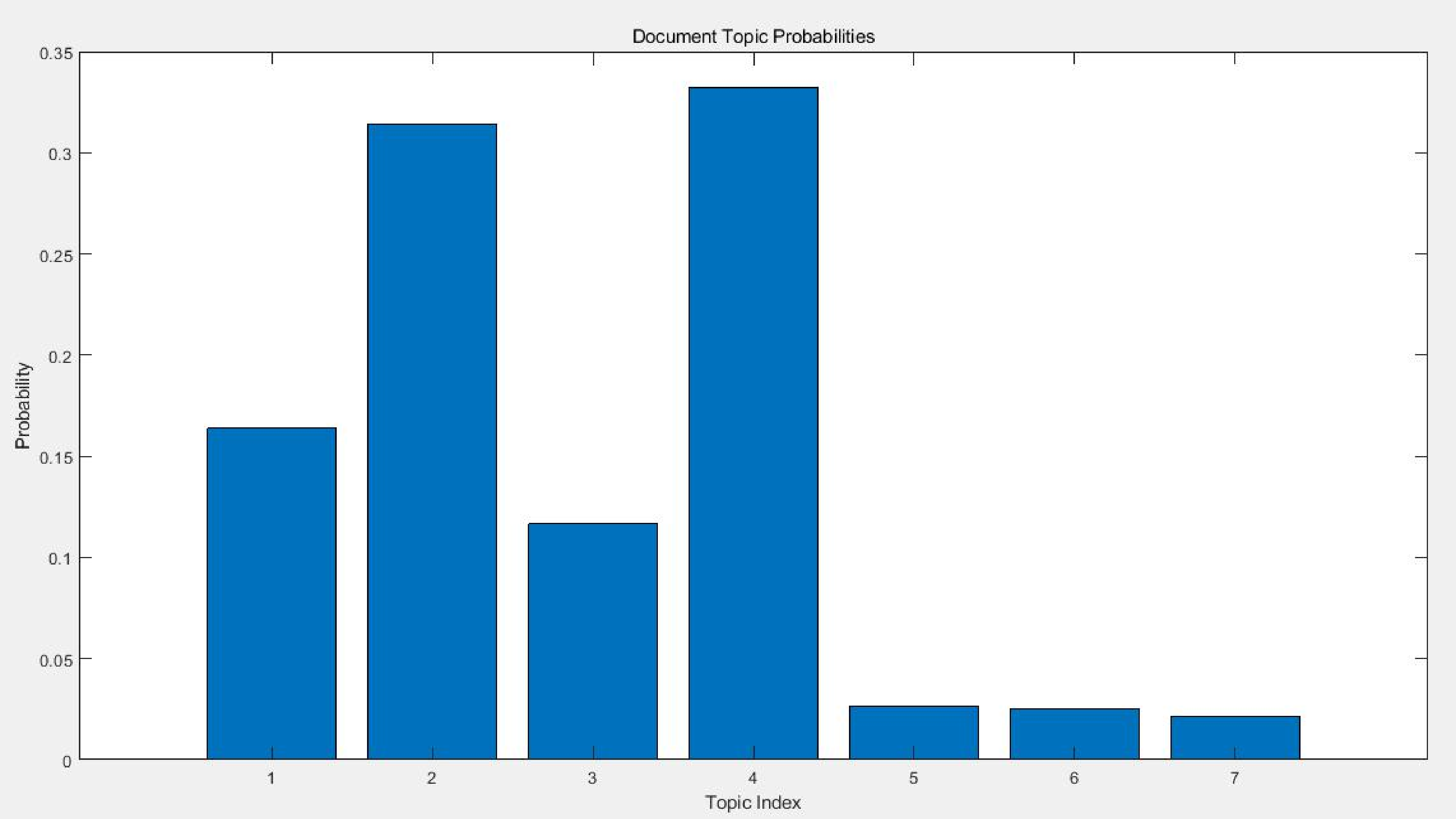
```
figure;
for topicIdx = 1:4
  subplot(2,2,topicIdx)
  wordcloud(mdl,topicIdx);
  title("Topic " + topicIdx)
end
```


Topic 1
bottom connect output software material sometimes motor assemble overheat stuck stick jammed jam assembled liquid sort assembler get fails bent time product leave classifier engine sorter mixer switch occasionally
Topic 2
freeze interface first motor once mixer blow program start begin fuse reel continue thing keep scanner blend item neglect bend material sorter spool trip crack assembly stick capacitor controller programming
Topic 3
underneath yield collect out constructing capacitor show wear panel burn supply appear power twist fail sign agent break give crack return fine cool failure item construct assembler roller control construction defect turn back backup
Topic 4
underneath assembly shut unexpectedly extreme pipe yield burst overheats scanner software blender mix leak place spill produce floor mixer cool crash mixing coolant leakage freeze spray frozen down everywhere

View Mixtures of Topics in Documents

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

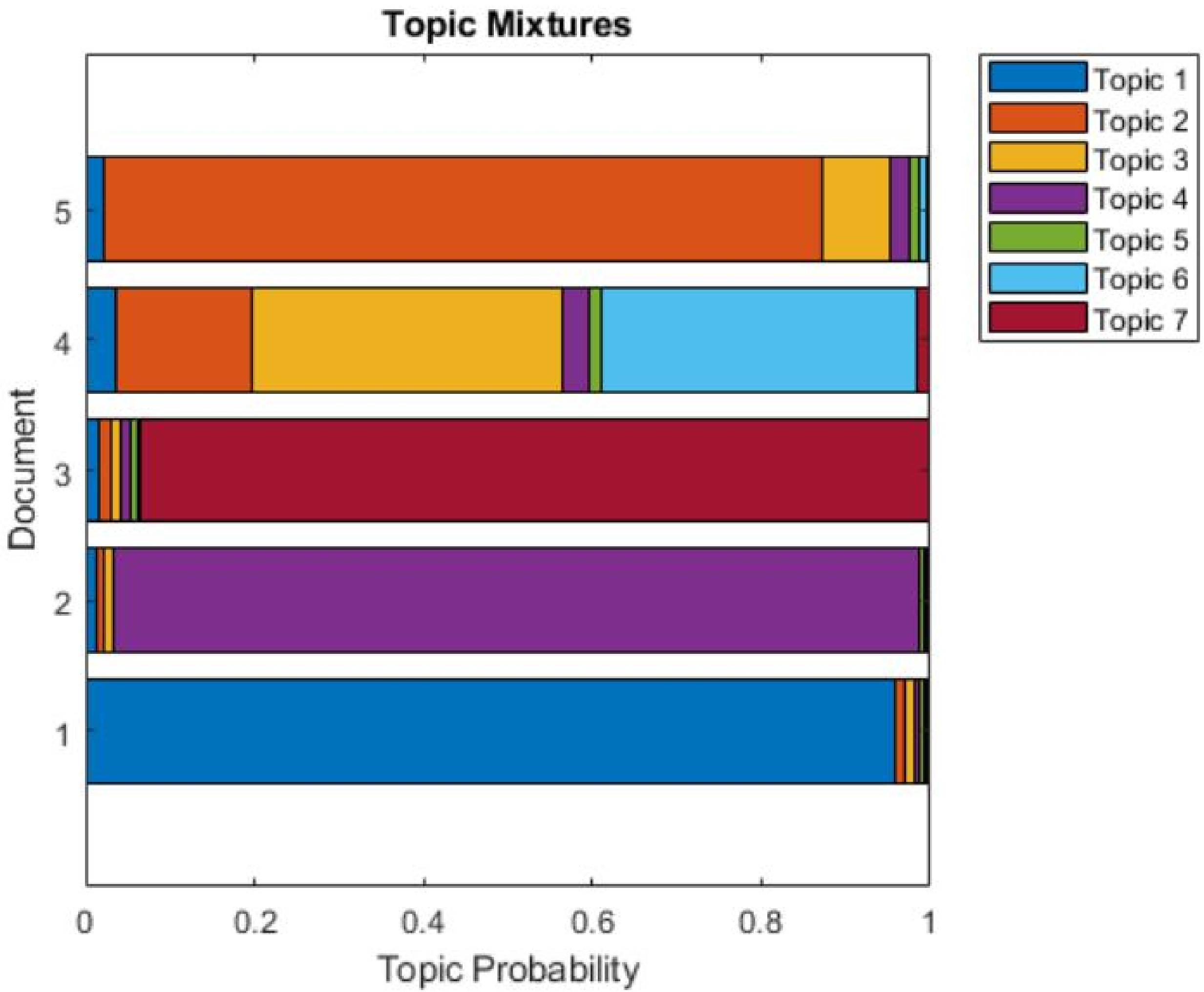
```
newDocument = tokenizedDocument("Coolant is pooling underneath sorter.");
topicMixture = transform(mdl,newDocument);
figure
bar(topicMixture)
xlabel("Topic Index")
ylabel("Probability")
title("Document Topic Probabilities")
```


Document Topic Probabilities
Probability
Topic Index
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
1
2
3
4
5
6
7

View Mixtures of Topics in Documents

- Visualize multiple topic mixtures using stacked bar charts. Visualize the topic mixtures of the first 5 input documents.

```
figure
topicMixtures = transform(mdl,documents(1:5));
barh(topicMixtures(1:5,:),'stacked')
xlim([0 1])
title("Topic Mixtures")
xlabel("Topic Probability")
ylabel("Document")
legend("Topic " + string(1:numTopics),'Location','northeastoutside')
```



Topic Mixtures
Document
5
4
3
2
1
0
0.2
0.4
0.6
0.8
1
Topic Probability
Topic 1
Topic 2
Topic 3
Topic 4
Topic 5
Topic 6
Topic 7

TY OF CHINA