WHITE PAPER

Getting Started with Text Analytics in MATLAB

MathWorks®

Introduction

If you are familiar with typical data analytics, trying text analytics might be simpler than you expect. The steps for a text analytics workflow aren't unique to text: Access and explore the data, preprocess it, build a model, and share the results or the model. However, working with text data often raises some questions:

- Where does the data come from?
- What is it used for?
- Can any of the processing be automated?
- How is a model created with text data?

Text data is surprisingly accessible. Engineers and scientists generate a lot of text data as part of day-to-day operations. This data comes from internal sources such as internal reports, maintenance logs, work orders, and technical support cases. Text data can provide important information such as cause of equipment failure, pain points in products and process design, and action recommendations based on historic data.

There are also many external sources of data, such as information in social media, news, blogs, forums, and other platforms. This data can be valuable for getting timely insights into gaps and opportunities in scientific research; market intelligence for improving product/process design; and economic or policy information in forecasting models for product demand.

In each of these cases, text analytics with MATLAB® can be useful in automating the process of extracting information from text, significantly reducing the time required for manual processing.

This white paper highlights common text analytics applications followed by a typical workflow and some examples to get you started with exploring and building models with your own text dataset.

Text Analytics Applications

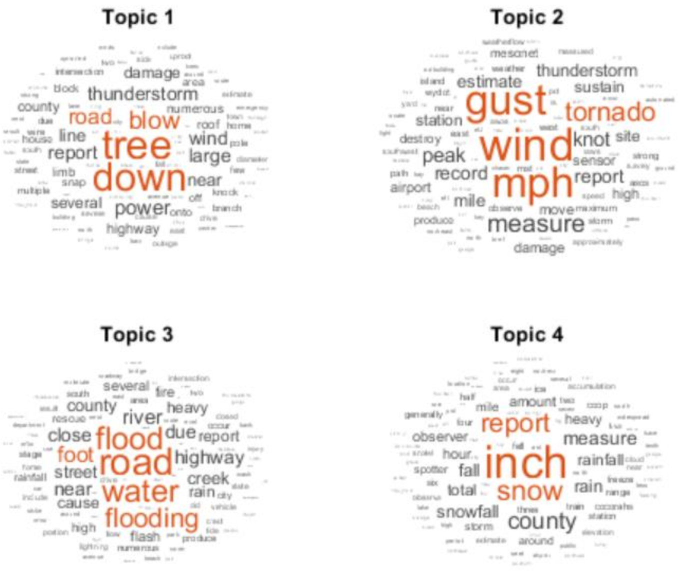
Text analytics is the process of uncovering hidden patterns from raw human language to enable better decision-making and predictions.

This section goes through four real-world applications of text analytics:

- Topic Modeling
- Classification
- Sentiment Analysis
- Summarization

Topic Modeling

Identify topics that reflect underlying patterns and relationships from raw text data in a collection of documents.

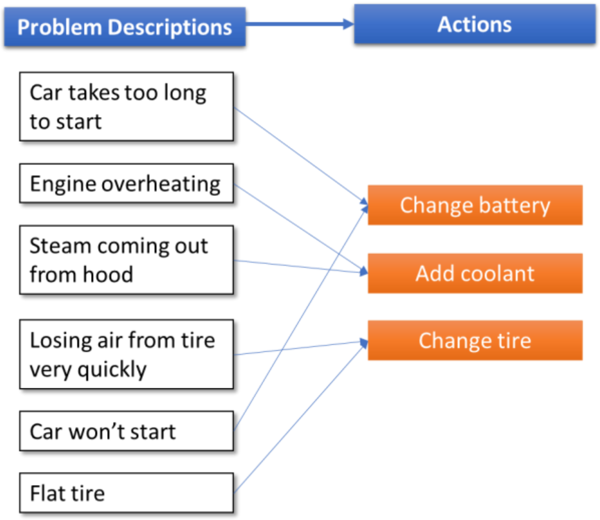


Real-world applications:

- Get insight into underlying causes of damages or failures and their correlations using maintenance and repair reports and work orders for manufacturing equipment, cars, or aircraft.
- Identify major pain points or gaps for informing product or process design teams with customer surveys, reviews from social media, blogs, forums, and customer and technical support cases.

Classification

Classify documents into predetermined categories for efficient information retrieval and prediction.

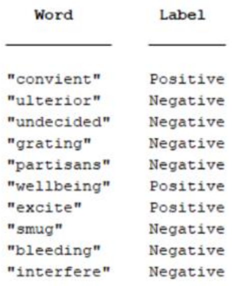


Real-world applications:

- Recommend actions or repair work needed based on text describing malfunctions or faults using a predictive model built with historical data.
- Detect fraudulent reports using features extracted from text for cybersecurity or for asset selection using a prebuilt classification model.

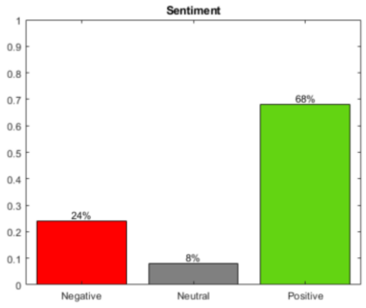
Sentiment Analysis

Identify and score sentiments expressed in text.



```
    Word            Label
 ___________     ________

 "convient"      Positive
 "ulterior"      Negative
 "undecided"     Negative
 "grating"       Negative
 "partisans"     Negative
 "wellbeing"     Positive
 "excite"        Positive
 "smug"          Negative
 "bleeding"      Negative
 "interfere"     Negative
```

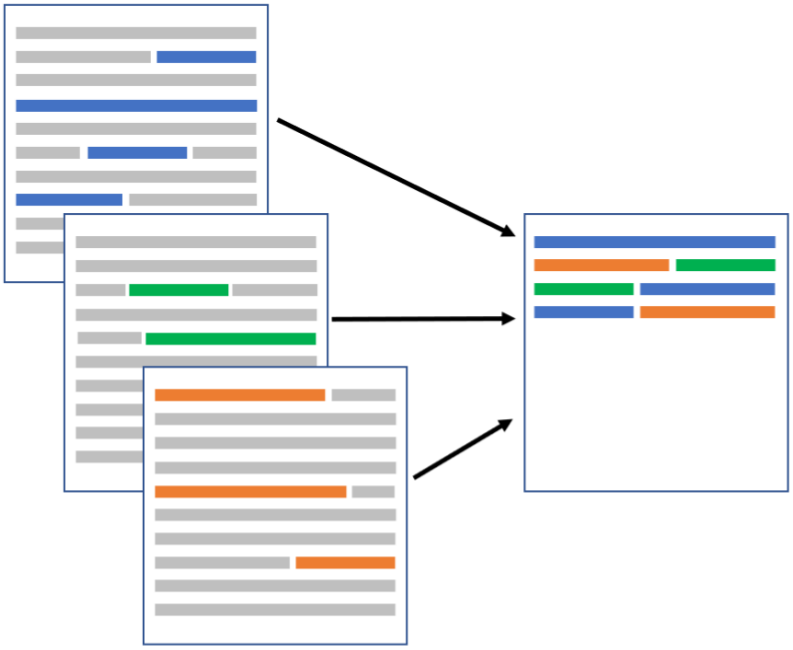


Real-world applications:

- Identify pain points and gaps for better product or process design using sentiment scores derived from customer surveys and social media.
- Build an asset selection model for trading using sentiment scores of financial reports and news articles.

Summarization

Extract a summary from one or more documents automatically.



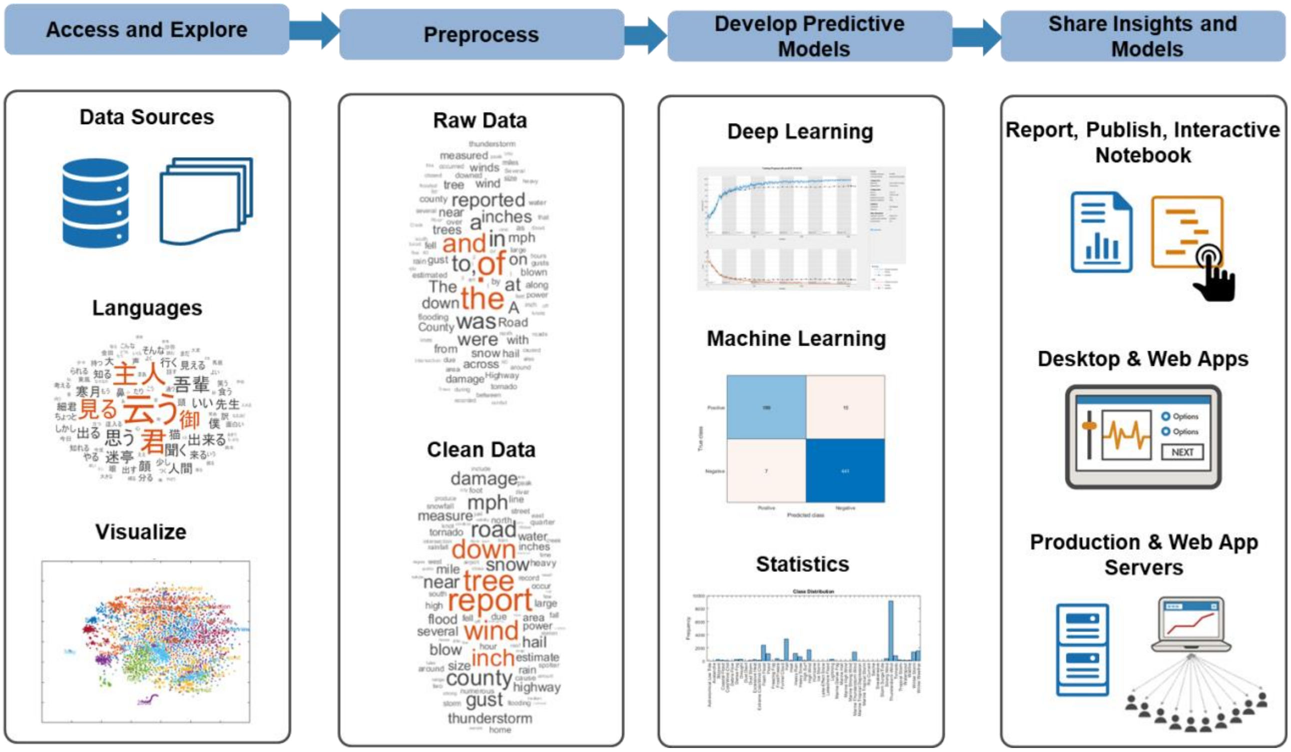
Real-world application:

- Identify opportunities and gaps in scientific research by summarizing technical articles.
- Highlight and understand relevant information faster by summarizing internal reports.

Access and Preprocess Text Data

An end-to-end text analytics workflow involves the following four steps:

1. Access data from databases, the web, and internal file repositories and explore by visualization.
2. Preprocess data by eliminating extraneous information such as punctuation, common words, or stop words such as “a” and “the.”
3. Build predictive models by using machine or deep learning algorithms.
4. Share insights and use predictive models in applications.

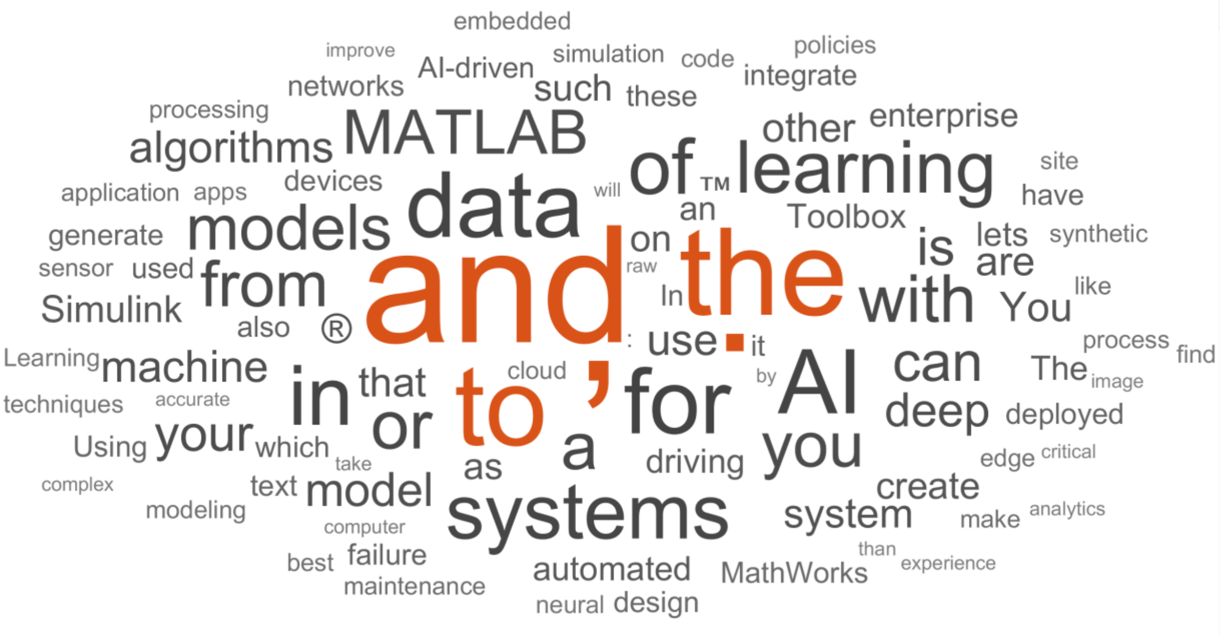


If this looks complicated, don’t worry. Examples will show you how to access data, preprocess text, and build text analytics models. Here’s an example to get you started with accessing and exploring text data.

Get Started by Accessing and Exploring Text

You may have text data in various formats such as Microsoft® Word® document, PDF, plain text, Microsoft Excel, databases, and web pages. This example demonstrates reading in a page on artificial intelligence from the MathWorks web site and seeing what words are used using simple visualizations.

```
url = "https://www.mathworks.com/discovery/artificial-intelligence.html";
code = webread(url); % read the whole page
tree = htmlTree(code); % let’s take a look at the structure of the page
subtree = findElement(tree, "p"); % find the paragraph elements
orgtext = extractHTMLText(subtree); % extract the text from paragraph elements
text = tokenizedDocument(orgtext); % break it down into words
wordcloud(text)
```



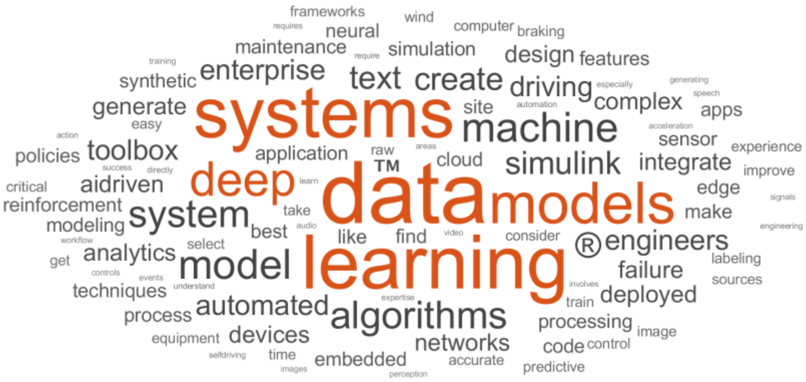
Preprocessing Text Data

You should have noticed that there are some issues with the word cloud, in particular the inclusion of punctuation, symbols, and words like “and”, “the,” and “to” that are not likely to add value. These low-information words are called **stop words**.

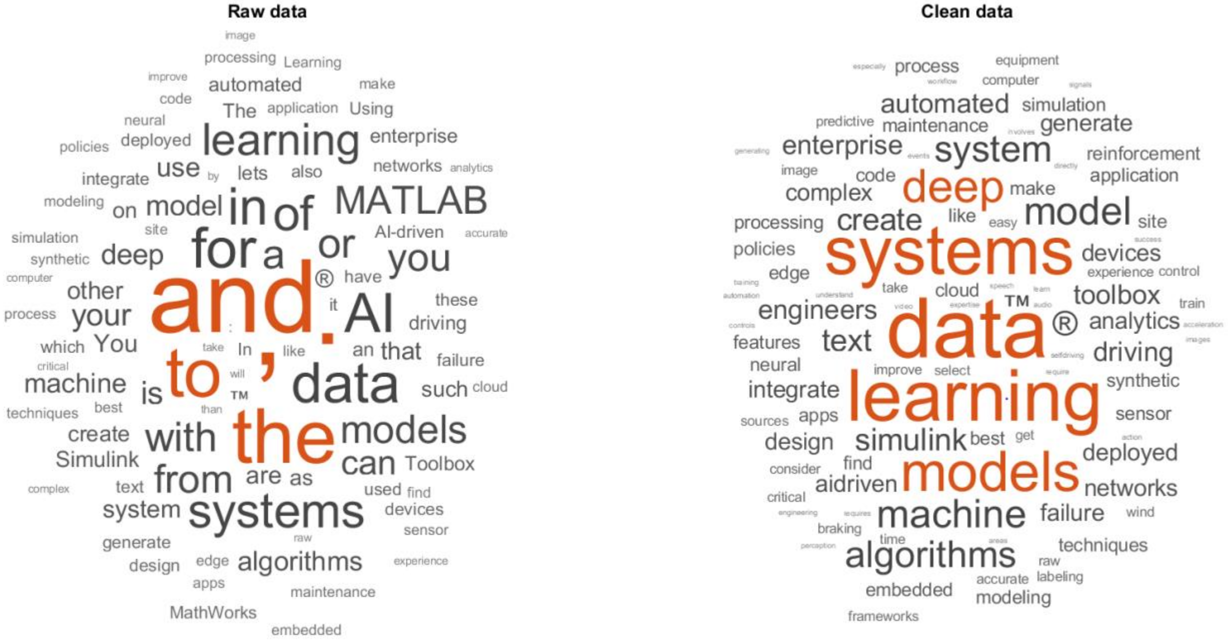
Data cleaning is an important step in trying to extract information from the data. Plot the data as a word cloud to see what effects the cleaning has had and whether further data cleaning is necessary.

To clean the data and plot another word cloud, try:

```
cleanText = lower(text); % make text lower case
cleanText = tokenizedDocument(cleanText); % split text into individual words
commonWords = ["ai", "artificial", "intelligence", "matlab", "simulink",
"mathworks"];
cleanText = removeWords(cleanText, commonWords); % remove words that  won’t
give us much information
cleanText = removeStopWords(cleanText); % remove stop words such as "a", "the"
cleanText = erasePunctuation(cleanText); % remove punctuation
wordcloud(cleanText) % plot wordcloud again
```



```
subplot(1, 2, 1), wordcloud(text); title('Raw data') % plot word clouds side-
by-side for comparison
subplot(1, 2, 2), wordcloud(cleanText); title('Clean data')
```



The clean word cloud relays more information about what is on the page. Data cleaning can often be the most time-consuming part of data analytics. However, it gets easier with experience and knowledge of common data preprocessing.

Learn more about common text data preprocessing techniques.

Sometimes, all you need is simple “string” manipulation techniques. MATLAB includes these useful functions to:

- Search for specific strings or characters (`regexp` and `strfind`)
- Replace certain words (`regexprep` and `replaceWords`)
- Look at certain sections of a document (`extractBetween`)
- Compare strings (`strcmp`)
- Count how many times certain words are mentioned in a document (`count`)

See a list of string manipulation functions.

Text Summarization

Another way to look at the text data is to extract a summary. To create a six-sentence summary, use:

```
orgLines = splitSentences(join(orgtext)); % Join all the paragraphs and split
into sentences
tok_orgLines = tokenizedDocument(orgLines); % Tokenize the sentences
summary = extractSummary(tok_orgLines, 'SummarySize', 6, 'OrderBy',"position");
% Look at a 6-sentence summary
strSummary = join(joinWords(summary))
```

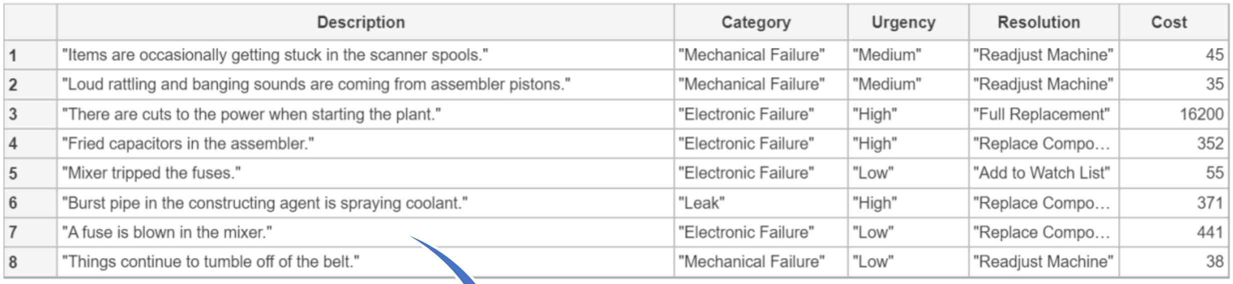
```
"Beyond automated driving , AI is also used in models that predict machine
failure , indicating when they will require maintenance ; health and sensor
analytics such as patient monitoring systems ; and robotic systems that
learn and improve directly from experience .
Data preparation requires domain expertise , such as experience in speech
and audio signals , navigation and sensor fusion , image and video
processing , and radar and lidar .
In automated driving systems , AI for perception must integrate with
algorithms for localization and path planning and controls for braking ,
acceleration , and turning .
AI models need to be deployed to CPUs , GPUs , and / or FPGAs in your final
product , whether part of an embedded or edge device , enterprise system ,
or cloud .
Statistics and Machine Learning Toolbox ™ makes the hard parts of machine
learning easy with apps for training and comparing models , advanced signal
processing and feature extraction , classification , regression , and
clustering algorithms for supervised and unsupervised learning .
For example , in an automated driving system , you use AI and simulation to
design the controller for braking , acceleration , and turning ."
```

Do these sentences describe what the page is about? Yes. These are somewhat long sentences, but they capture the main points of the article.

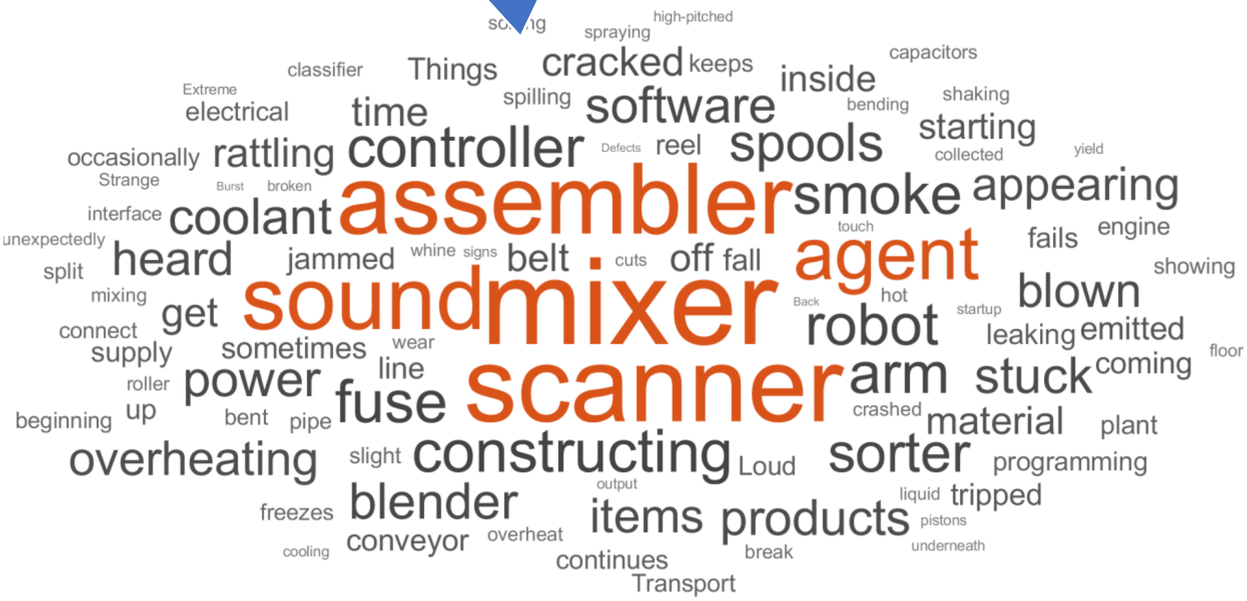
Build Predictive Models

One common application of text analytics is finding hidden patterns in the data. If you can identify the patterns, you can take corrective actions to resolve the issues. Here’s a dataset that lists different categories of factory reports with written descriptions of the events.

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



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

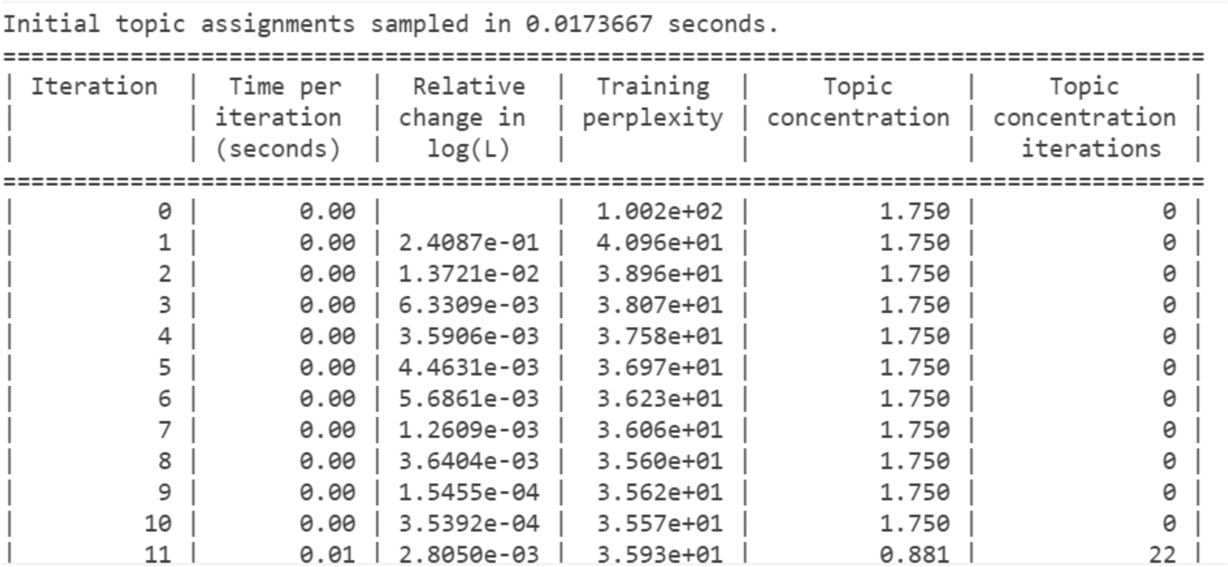


Once the data is imported and cleaned (similar to the previous section), the next step in building a model is converting the text into numeric form. You can use the bag-of-words modeling approach to create a matrix of words with their frequencies.

```
bag = bagOfWords(documents); % documents are tokenized and cleaned data
bag = removeInfrequentWords(bag,2); % remove infrequent words
bag = removeEmptyDocuments(bag); % remove empty documents
```

Next, use the latent Dirichlet allocation (LDA) method to uncover hidden topics in the dataset. The `fitlda` function call will fit an LDA model to the matrix of words and their frequencies.

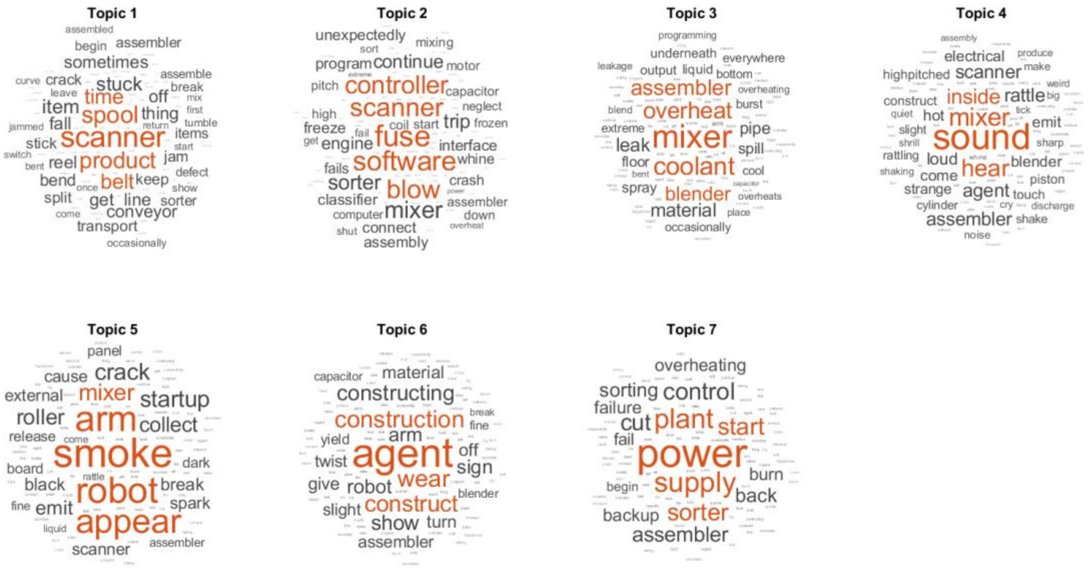
```
numTopics = 7; % assumed number of topics
mdl = fitlda(bag,numTopics);
```



```
Initial topic assignments sampled in 0.0173667 seconds.
=====================================================================================
| Iteration  |  Time per  |  Relative  |  Training  |     Topic     |     Topic     |
|            |  iteration | change in  | perplexity | concentration | concentration |
|            |  (seconds) |   log(L)   |            |               |   iterations  |
=====================================================================================
|          0 |       0.00 |            |  1.002e+02 |         1.750 |             0 |
|          1 |       0.00 | 2.4087e-01 |  4.096e+01 |         1.750 |             0 |
|          2 |       0.00 | 1.3721e-02 |  3.896e+01 |         1.750 |             0 |
|          3 |       0.00 | 6.3309e-03 |  3.807e+01 |         1.750 |             0 |
|          4 |       0.00 | 3.5906e-03 |  3.758e+01 |         1.750 |             0 |
|          5 |       0.00 | 4.4631e-03 |  3.697e+01 |         1.750 |             0 |
|          6 |       0.00 | 5.6861e-03 |  3.623e+01 |         1.750 |             0 |
|          7 |       0.00 | 1.2609e-03 |  3.606e+01 |         1.750 |             0 |
|          8 |       0.00 | 3.6404e-03 |  3.560e+01 |         1.750 |             0 |
|          9 |       0.00 | 1.5455e-04 |  3.562e+01 |         1.750 |             0 |
|         10 |       0.00 | 3.5392e-04 |  3.557e+01 |         1.750 |             0 |
|         11 |       0.01 | 2.8050e-03 |  3.593e+01 |         0.881 |            22 |
```

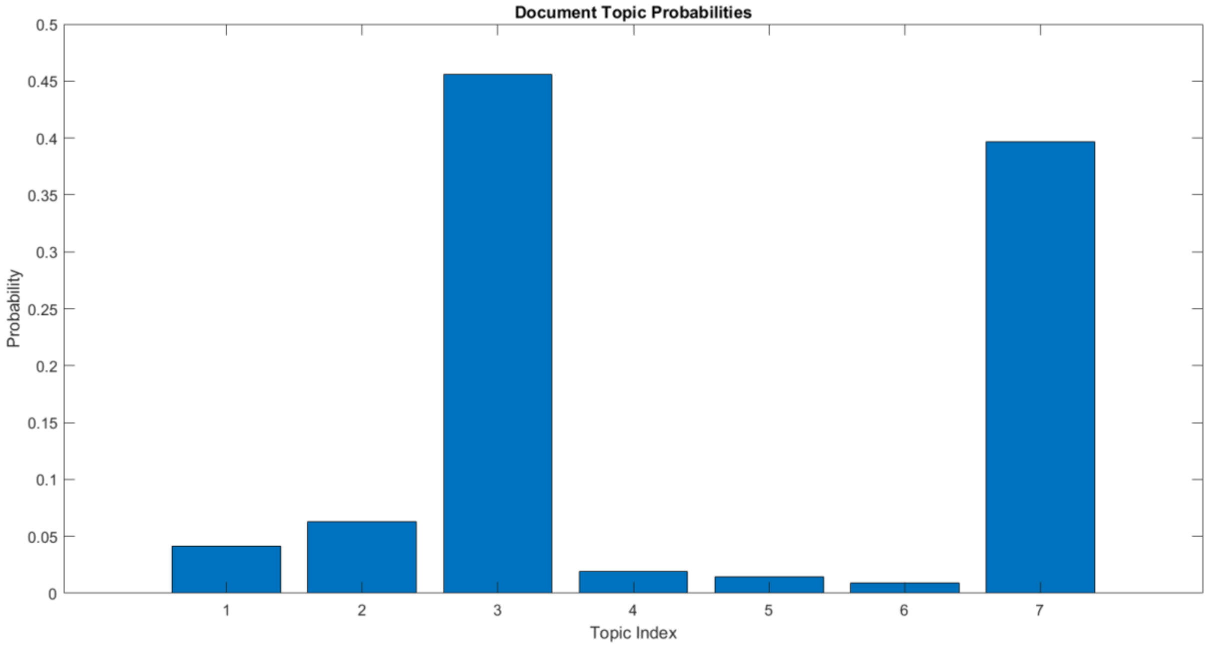
In this example, assume there are seven topics in the data (often you might not know how many topics to expect). One approach to decide on the number of topics is to evaluate the goodness-of-fit of the model using perplexity of the validation set. See an example.

Next, visualize the words for each topic with a word cloud. You should be able to see that some patterns are starting to emerge from the data.


Topic 1
assembled begin assembler sometimes curve crack stuck assemble leave time off break mix item thing first jammed fall spool return tumble stick scanner items start switch reel product jam bent defect bend once belt keep show split get line sorter come conveyor transport occasionally
Topic 2
unexpectedly mixing sort program continue motor pitch controller capacitor high scanner neglect freeze coil start trip frozen fail get engine fuse interface fails software whine sorter blow crash classifier power assembler computer mixer down shut connect overheat assembly
Topic 3
programming underneath everywhere leakage output liquid bottom assembler overheating blend overheat burst extreme pipe leak mixer spill floor coolant cool bent spray capacitor blender overheats material place occasionally
Topic 4
assembly electrical produce highpitched scanner make weird construct inside rattle big quiet hot mixer tick slight emit shrill sound sharp rattling loud blender shaking come whine hear piston strange agent touch cylinder cry discharge assembler shake noise
Topic 5
panel cause crack external mixer startup roller arm collect release come board smoke dark black rattle break robot fine emit spark liquid appear scanner assembler
Topic 6
capacitor material constructing construction break fine yield arm off twist agent sign give robot wear blender slight construct show turn assembler
Topic 7
overheating sorting control failure cut plant start fail power burn begin supply back backup sorter assembler

Test the model with a new narrative and see if it correctly identified the topic.

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



According to the model, the new narrative is a mixture of Topics 3 and 7 (related to coolant and sorter). By examining Topics 3 and 7 word clouds, you can verify that the model’s prediction is correct. You can then automatically alert the response team about the issue that needs attention. This approach will eliminate the need for a person to manually inspect each incident to take corrective actions. If you have location data available and find a correlation between location and certain topics, that may alert you to a bigger problem with infrastructure.

There are many possibilities and ways to take advantage of the data to ultimately make better decisions. The data you need to build these models is probably already out there. You may just need to look into it a bit more closely with tools like MATLAB.

Share Insights and Use Predictive Models in Applications

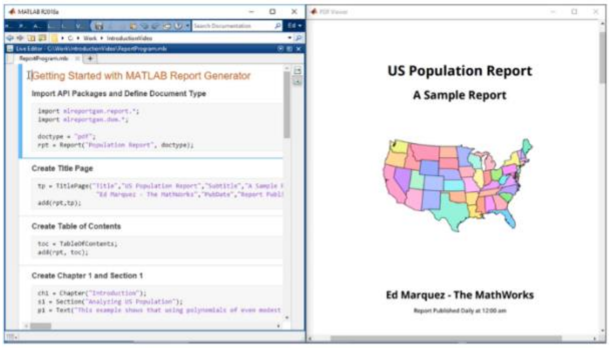
Once you have built a model using text analytics and validated it for acceptable performance, there are several ways that you can share the results and models with your team or management.

Share an interactive Live Editor notebook with MATLAB users or publish live scripts as HTML, PDF, LaTeX, or Microsoft Word to share with non-MATLAB users.

How to Publish MATLAB Code

Share your work by publishing MATLAB® code from the MATLAB Editor to HTML and other formats.

Generate formatted reports with results and figures in PDF, Microsoft Word or PowerPoint, or HTML using MATLAB Report Generator.


MATLAB R2018a
Search Documentation
C: ▸ Work ▸ IntroductionVideo
Live Editor - C:\Work\IntroductionVideo\ReportProgram.mlx
ReportProgram.mlx
PDF Viewer

Getting Started with MATLAB Report Generator

Import API Packages and Define Document Type

```
import mlreportgen.report.*;
import mlreportgen.dom.*;

doctype = "pdf";
rpt = Report("Population Report", doctype);
```

Create Title Page

```
tp = TitlePage("Title","US Population Report","Subtitle","A Sample R
              "Ed Marquez - The MathWorks","PubDate","Report Publi
add(rpt,tp);
```

Create Table of Contents

```
toc = TableOfContents;
add(rpt, toc);
```

Create Chapter 1 and Section 1

```
ch1 = Chapter("Introduction");
s1 = Section("Analyzing US Population");
p1 = Text("This example shows that using polynomials of even modest
```

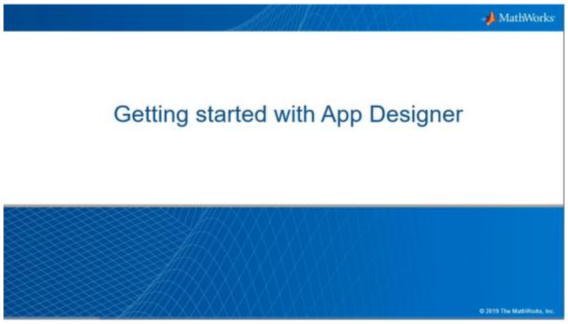
US Population Report

A Sample Report

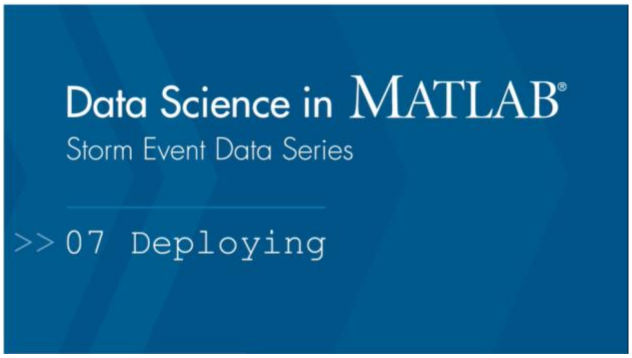
Ed Marquez - The MathWorks

Report Published Daily at 12:00 am

Create a standalone or web app using MATLAB App Designer.



Host the application on a production server or web app server



More resources that you can use to keep learning:

- Text Analytics Toolbox - Overview
- Text Analytics Toolbox - Documentation
- 8 MATLAB Cheat Sheets for Data Science - Cheat Sheets
- Text Analytics in MATLAB (23:36) - Video

Download a free trial