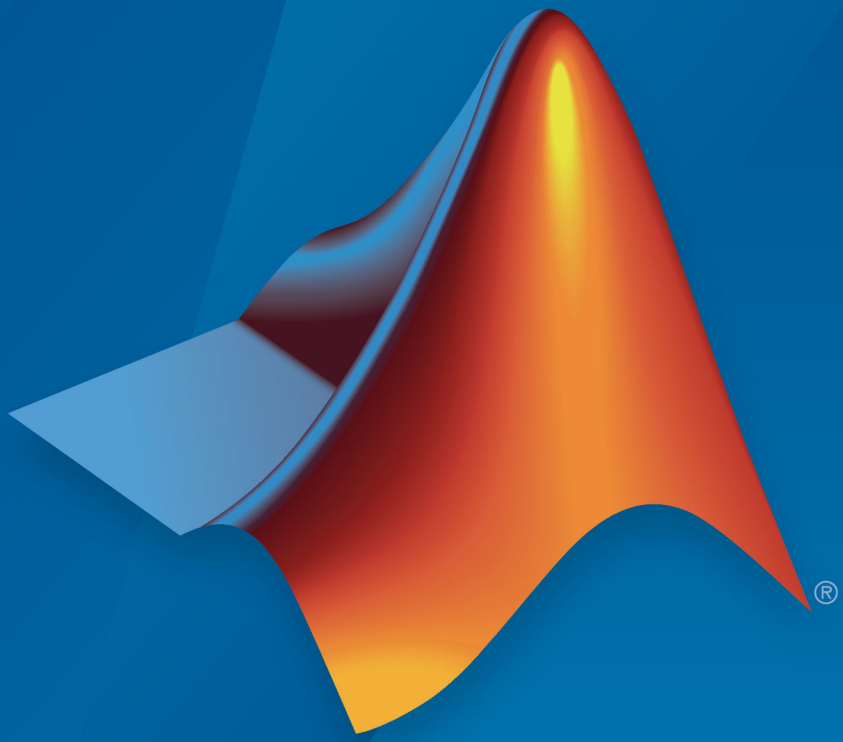
Text Analytics Toolbox™ Release Notes



MATLAB®

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™ Release Notes

Contents

R2017b

R2017b

Version: 1.0

New Features

Text Preprocessing: Prepare text for analysis by automatically extracting and prepocessing words from raw text

You can perform the following common character level preprocessing steps to prepare text data before splitting it into words:

- Erase HTML and XML tags using `eraseTags`.
- Erase URLs using `eraseURLs`.
- Erase punctuation using `erasePunctuation`.
- Convert HTML and XML entities into characters using `decodeHTMLEntities`.

After character level preprocessing, you can split text into words using `tokenizedDocument` which creates an array of `tokenizedDocument` objects. With a `tokenizedDocument` array, you can perform the following word level preprocessing steps:

- Remove specified words from an array of documents using `removeWords`.
- Remove a common list of stop words which are not useful for analysis (such as "a" and "the") using `removeWords` and `stopWords`.
- Remove long and short words using `removeLongWords` and `removeShortWords` respectively.
- Stem words using `normalizeWords`.

For an example showing how to preprocess text data and prepare for it for analysis, see “Prepare Text Data for Analysis”.

Machine Learning Algorithms: Discover topics and clusters of documents using Latent Dirichlet Allocation (LDA) and Latent Semantic Analysis (LSA)

You can analyze text data using the Latent Dirichlet Allocation topic model. Latent Dirichlet Allocation models a collection of documents as mixtures of topics.

Fit an `ldaModel` using `fitlda`. You can resume training using `resume`. Using `ldaModel` objects, you can perform the following tasks:

- Visualize topics and word importance of an LDA model using `wordcloud` and `topkwords`.

- Extract features, or reduce dimensionality using `transform`. This function transforms documents into the lower dimensional topic probability space.
- Predict top topics of documents using `predict`.
- Calculate document log probabilities and detect outliers using `logp`.

You can also use Latent Semantic Analysis to model your text data.

Fit an `lsaModel` using `fitlsa`. To use an LSA model as a feature extractor, or a dimension reducing tool, use `transform`. This function transforms documents into a lower dimensional semantic space.

For an example showing how to use LDA to analyze text data, see "Analyze Text Data Using Topic Models". For more information on LSA models, see `lsaModel`.

Word Embeddings: Convert words to numeric vectors using word2vec, FastText, and GloVe word embedding models

Use word embeddings to discover relationships between words. Word embeddings model words as vectors in a fixed dimensional space. For example, a word embedding may learn the relationship "king" – "man" + "woman" = "queen".

Create a `WordEmbedding` object by using one of the following methods:

- Import word embedding files from word2vec, FastText, and GloVe using `readWordEmbedding`.
- Train your own word embeddings from text data using `trainWordEmbedding`.

With a `WordEmbedding` object, you can do the following:

- Map words to vectors and back using `word2vec` and `vec2word`.
- Write the word embedding to a file using `writeWordEmbedding`.

For an example showing how to explore word embeddings, see "Visualize Word Embeddings Using Text Scatter Plots".

Text Plots: Visualize text data using word clouds and text scatter plots

Text Analytics Toolbox extends the functionality of the `wordcloud` (MATLAB®) function. It adds support for the following tasks:

- Create word clouds directly from string. `wordcloud` automatically tokenizes, preprocesses, and counts word frequencies of string input.
- Create word clouds from bag-of-words models.
- Create word clouds from LDA topics.

You can also visualize text data using 2-D and 3-D text scatter plots. Use `textscatter` and `textscatter3` to plot words at specified coordinates of 2-D and 3-D scatter plots respectively.

For an example showing how to visualize collections of text data using word clouds, see "Visualize Text Data Using Word Clouds".

Document Import: Read text from PDF and Microsoft Word files

You can extract text data directly from plain text, PDF, and Microsoft® Word files using `extractFileText`.

For an example showing how to extract text data from files and import it into MATLAB, see "Extract Text Data From Files".

Text Statistics: Calculate word frequency and TF-IDF matrices from document collections

A bag-of-words model (also known as a term-frequency counter) records the number of times that words appear in each document of a collection.

Create a `bagOfWords` object using `bagOfWords`.

With a `bagOfWords` object, you can perform the following tasks:

- Encode documents as a matrix of word counts using `encode`.
- View the most frequent words using `topkwords`.
- Add and remove documents using `addDocument` and `removeDocument` respectively.
- Remove empty documents using `removeEmptyDocuments`.
- Remove infrequent words using `removeInfrequentWords`.

You can input `bagOfWords` objects directly into `fitlda`, `fitlsa`, and `wordcloud`.

You can create tf-idf matrices from a bag-of-words model using `tfidf`. A tf-idf matrix is a statistic that captures word importance in a collection of documents. It captures the number of times each word appear in a collection, and how many documents each word appears in.

For more information, see `bagOfWords`.

Word Normalization: Convert words to their word roots using the Porter stemming algorithm

To group different forms of English words by reducing them to a common stem, use `normalizeWords`. For example, use this function to reduce the words "walk", "walks", "walking" and "walk" all to their word root "walk". `normalizeWords` uses the Porter stemmer.

For more information, see `normalizeWords`.